account.createTheme	Create a theme
account.installTheme	Install a theme
account.saveTheme	Save a theme
account.getTheme	Get theme information

Working with contacts and top peers

Name	Description
account.getContactSignUpNotification	Whether the user will receive notifications when contacts sign up
account.setContactSignUpNotification	Toggle contact sign up notifications
contacts.acceptContact	If the peer settings of a new user allow us to add him as contact, add that user as contact
contacts.addContact	Add an existing telegram user as contact
contacts.block	Adds the user to the blacklist.
contacts.deleteByPhones	Delete contacts by phone number
contacts.deleteContacts	Deletes several contacts from the list.
contacts.getBlocked	Returns the list of blocked users.
contacts.getContactIDs	Get contact by telegram IDs
contacts.getContacts	Returns the current user's contact list.
contacts.getLocated	Get contacts near you
contacts.getSaved	Get all contacts
contacts.getStatuses	Returns the list of contact statuses.
contacts.getTopPeers	Get most used peers
contacts.importContacts	Imports contacts: saves a full list on the server, adds already registered contacts to the contact list, returns added contacts and their info.
contacts.resetSaved	Delete saved contacts
contacts.resetTopPeerRating	Reset rating of top peer
contacts.search	Returns users found by username substring.
contacts.toggleTopPeers	Enable/disable top peers
contacts.unblock	Deletes the user from the blacklist.

Working with dialogs

Name	Description
messages.getDialogs	Returns the current user dialog list.
messages.getPeerDialogs	Get dialog info of specified peers
messages.getPinnedDialogs	Get pinned dialogs
messages.toggleDialogPin	Pin/unpin a dialog
messages.reorderPinnedDialogs	Reorder pinned dialogs
messages.getDialogUnreadMarks	Get dialogs manually marked as unread
messages.markDialogUnread	Manually mark dialog as unread
messages.getPeerSettings	Get peer settings
messages.hidePeerSettingsBar	Should be called after the user hides the report spam/add as contact bar of a new chat, effectively prevents the user from executing the actions specified in the peer's settings.
messages.getOnlines	Get count of online users in a chat
messages.sendScreenshotNotification	Notify the other user in a private chat that a screenshot of the chat was taken
messages.setTyping	Sends a current user typing event (see SendMessageAction for all event types) to a conversation partner or group.

Working with drafts

Name	Description
messages.clearAllDrafts	Clear all drafts.
messages.getAllDrafts	Save get all message drafts.
messages.saveDraft	Save a message draft associated to a chat.

Working with emoji keywords

Name	Description
messages.getEmojiKeywords	Get localized emoji keywords
messages.getEmojiKeywordsDifference	Get changed emoji keywords
messages.getEmojiKeywordsLanguages	Get info about an emoji keyword localization
messages.getEmojiURL	Returns an HTTP URL which can be used to automatically log in into translation platform and suggest new emoji replacements. The URL will be valid for 30 seconds after generation

Working with folders

Name	Description
folders.deleteFolder	Delete a folder
folders.editPeerFolders	Edit peers in folder

Working with games

Working with games

Name	Description
messages.getGameHighScores	Get highscores of a game
messages.getInlineGameHighScores	Get highscores of a game sent using an inline bot
messages.setGameScore	Use this method to set the score of the specified user in a game sent as a normal message (bots only).
messages.setInlineGameScore	Use this method to set the score of the specified user in a game sent as an inline message (bots only).

Working with localization packs

Name	Description
langpack.getDifference	Get new strings in languagepack
langpack.getLangPack	Get localization pack strings
langpack.getLanguage	Get information about a language in a localization pack
langpack.getLanguages	Get information about all languages in a localization pack
langpack.getStrings	Get strings from a language pack

Working with media autodownload settings

Name	Description
account.getAutoDownloadSettings	Get media autodownload settings
account.saveAutoDownloadSettings	Change media autodownload settings

Working with message reactions

Name	Description
messages.getMessagesReactions	Get message reactions
messages.sendReaction	Send reaction to message
messages.getMessageReactionsList	Get full message reaction list

Working with messages

Name	Description
messages.deleteHistory	Deletes communication history.
messages.deleteMessages	Deletes messages by their identifiers.
messages.editMessage	Edit message
messages.forwardMessages	Forwards messages by their IDs.
messages.getHistory	Gets back the conversation history with one interlocutor / within a chat
messages.getMessageEditData	Find out if a media message's caption can be edited
messages.getMessages	Returns the list of messages by their IDs.
messages.getMessagesViews	Get and increase the view counter of a message sent or forwarded from a channel
messages.getRecentLocations	Get live location history of a certain user
messages.getSearchCounters	Get the number of results that would be found by a messages.search call with the same parameters
messages.getUnreadMentions	Get unread messages where we were mentioned
messages.readHistory	Marks message history as read.
messages.readMentions	Mark mentions as read
messages.readMessageContents	Notifies the sender about the recipient having listened a voice message or watched a video.
messages.receivedMessages	Confirms receipt of messages by a client, cancels PUSH-notification sending.
messages.search	Gets back found messages
messages.searchGlobal	Search for messages and peers globally
messages.sendMedia	Send a media
messages.sendMessage	Sends a message to a chat
messages.sendMultiMedia	Send an album of media
messages.updatePinnedMessage	Pin a message

Working with notification settings

Name	Description
account.registerDevice	Register device to receive PUSH notifications
account.unregisterDevice	Deletes a device by its token, stops sending PUSH-notifications to it.
account.updateDeviceLocked	When client-side passcode lock feature is enabled, will not show message texts in incoming PUSH notifications.
account.getNotifyExceptions	Returns list of chats with non-default notification settings
account.getNotifySettings	Gets current notification settings for a given user/group, from all users/all groups.
account.updateNotifySettings	Edits notification settings from a given user/group, from all users/all groups.
account.resetNotifySettings	Resets all notification settings from users and groups.

Working with other users

Name	Description
users.getFullUser	Returns extended user info by ID.

users.getUsers	Get basic info about a list of users

Working with payments

Name	Description
payments.getSavedInfo	Get saved payment information
payments.clearSavedInfo	Clear saved payment information
payments.getPaymentForm	Get a payment form
payments.validateRequestedInfo	Submit requested order information for validation
messages.setBotShippingResults	If you sent an invoice requesting a shipping address and the parameter is_flexible was specified, the bot will receive an updateBotShippingQuery update. Use this method to reply to shipping queries.
account.getTmpPassword	Get temporary payment password
payments.sendPaymentForm	Send compiled payment form
messages.setBotPrecheckoutResults	Once the user has confirmed their payment and shipping details, the bot receives an updateBotPrecheckoutQuery update. Use this method to respond to such pre-checkout queries. **Note**: Telegram must receive an answer within 10 seconds after the pre-checkout query was sent.
payments.getPaymentReceipt	Get payment receipt

Working with polls

Name	Description
messages.getPollResults	Get poll results
messages.sendVote	Vote in a poll

Working with scheduled messages

Name	Description
messages.sendScheduledMessages	Send scheduled messages right away
messages.getScheduledHistory	Get scheduled messages
messages.deleteScheduledMessages	Delete scheduled messages
messages.getScheduledMessages	Get scheduled messages

Working with sponsored proxies

Name	Description
help.getProxyData	Get promotion info of the currently-used MTProxy

Working with stickers

Name	Description
stickers.addStickerToSet	Add a sticker to a stickerset, bots only. The sticker set must have been created by the bot.
stickers.changeStickerPosition	Changes the absolute position of a sticker in the set to which it belongs; for bots only. The sticker set must have been created by the bot
stickers.createStickerSet	Create a stickerset, bots only.
stickers.removeStickerFromSet	Remove a sticker from the set where it belongs, bots only. The sticker set must have been created by the bot.
messages.clearRecentStickers	Clear recent stickers
messages.faveSticker	Mark a sticker as favorite
messages.getAllStickers	Get all installed stickers
messages.getArchivedStickers	Get all archived stickers
messages.getAttachedStickers	Get stickers attached to a photo or video
messages.getFavedStickers	Get faved stickers
messages.getFeaturedStickers	Get featured stickers
messages.getMaskStickers	Get installed mask stickers
messages.getRecentStickers	Get recent stickers
messages.getStickerSet	Get info about a stickerset
messages.getStickers	Get stickers by emoji
messages.saveRecentSticker	Add/remove sticker from recent stickers list
messages.installStickerSet	Install a stickerset
messages.readFeaturedStickers	Mark new featured stickers as read
messages.reorderStickerSets	Reorder installed stickersets
messages.searchStickerSets	Search for stickersets
messages.uninstallStickerSet	Uninstall a stickerset

Working with the user's account

Name	Description
account.changePhone	Change the phone number of the current account
account.confirmPhone	Confirm a phone number to cancel account deletion, for more info click here »
account.deleteAccount	Delete the user's account from the telegram servers. Can be used, for example, to delete the account of a user that provided the login code, but forgot the 2FA password and no recovery method is configured.

account.getAccountTTL	Get days to live of account
account.getPrivacy	Get privacy settings of current account
account.resetAuthorization	Log out an active authorized session by it hash
account.sendChangePhoneCode	Verify a new phone number to associate to the current account
account.sendConfirmPhoneCode	Send confirmation code to cancel account deletion, for more info click here »
account.setAccountTTL	Set account self-destruction period
account.setPrivacy	Change privacy settings of current account
account.updateProfile	Updates user profile.
account.updateStatus	Updates online user status.

Working with user profile pictures

Name	Description
photos.deletePhotos	Deletes profile photos.
photos.getUserPhotos	Returns the list of user photos.
photos.updateProfilePhoto	Installs a previously uploaded photo as a profile photo.
photos.uploadProfilePhoto	Updates current user profile photo.

Working with usernames

Name	Description
channels.checkUsername	Check if a username is free and can be assigned to a channel/supergroup
channels.updateUsername	Change the username of a supergroup/channel
account.updateUsername	Changes username for the current user.
account.checkUsername	Validates a username and checks availability.
contacts.resolveUsername	Resolve a @username to get peer info

Working with wallpapers

Name	Description
account.getWallPaper	Get info about a certain wallpaper
account.getWallPapers	Returns a list of available wallpapers.
account.installWallPaper	Install wallpaper
account.resetWallPapers	Delete installed wallpapers
account.saveWallPaper	Install/uninstall wallpaper
account.uploadWallPaper	Create and upload a new wallpaper

Schema

Below you will find the current TL-schema. More details on TL »

See also the detailed schema in JSON »

See also TL-Schema for end-to-end encrypted messages »

Layer 105

```
boolFalse#bc799737 = Bool;
boolTrue#997275b5 = Bool;

true#3fedd339 = True;

vector#1cb5c415 {t:Type} # [ t ] = Vector t;

error#c4b9f9bb code:int text:string = Error;

null#56730bcc = Null;

inputPeerEmpty#7f3b18ea = InputPeer;
inputPeerSelf#7da07ec9 = InputPeer;
inputPeerChat#179be863 chat_id:int = InputPeer;
inputPeerUser#7b8e7de6 user_id:int access_hash:long = InputPeer;
inputPeerChannel#20adaef8 channel_id:int access_hash:long = InputPeer;
inputPeerUserFromMessage#17bae2e6 peer:InputPeer msg_id:int user_id:int = InputPeer;
inputPeerChannelFromMessage#9c95f7bb peer:InputPeer msg_id:int channel_id:int = InputPeer;

inputUserEmpty#b98886cf = InputUser;
inputUserSelf#f7c1b13f = InputUser;
inputUser#d8292816 user_id:int access_hash:long = InputUser;
inputUserFromMessage#2d117597 peer:InputPeer msg_id:int user_id:int = InputUser;

inputPhoneContact#f392b7f4 client_id:long phone:string first_name:string last_name:string = InputContact;

inputFile#f52ff27f id:long parts:int name:string md5_checksum:string = InputFile;
inputFileBig#fa4f0bb5 id:long parts:int name:string = InputFile;

inputMediaEmpty#9664f57f = InputMedia;
inputMediaUploadedPhoto#1e287d04 flags:# file:InputFile stickers:flags.0?Vector<InputDocument> ttl_seconds:flag
inputMediaPhoto#b3ba0635 flags:# id:InputPhoto ttl_seconds:flags.0?int = InputMedia;
inputMediaGeoPoint#f9c44144 geo_point:InputGeoPoint = InputMedia;
inputMediaContact#f8ab7dfb phone_number:string first_name:string last_name:string vcard:string = InputMedia;
inputMediaUploadedDocument#5b38c6c1 flags:# nosound_video:flags.3?true file:InputFile thumb:flags.2?InputFile m
inputMediaDocument#23ab23d2 flags:# id:InputDocument ttl_seconds:flags.0?int = InputMedia;
inputMediaVenue#c13d1c11 geo_point:InputGeoPoint title:string address:string provider:string venue_id:string ve
inputMediaGifExternal#4843b0fd url:string q:string = InputMedia;
inputMediaPhotoExternal#e5bbfe1a flags:# url:string ttl_seconds:flags.0?int = InputMedia;
inputMediaDocumentExternal#fb52dc99 flags:# url:string ttl_seconds:flags.0?int = InputMedia;
inputMediaGame#d33f43f3 id:InputGame = InputMedia;
inputMediaInvoice#f4e096c3 flags:# title:string description:string photo:flags.0?InputWebDocument invoice:Invoi
inputMediaGeoLive#ce4e82fd flags:# stopped:flags.0?true geo_point:InputGeoPoint period:flags.1?int = InputMedia
inputMediaPoll#6b3765b poll:Poll = InputMedia;

inputChatPhotoEmpty#1ca48f57 = InputChatPhoto;
inputChatUploadedPhoto#927c55b4 file:InputFile = InputChatPhoto;
inputChatPhoto#8953ad37 id:InputPhoto = InputChatPhoto;

inputGeoPointEmpty#e4c123d6 = InputGeoPoint;
inputGeoPoint#f3b7acc9 lat:double long:double = InputGeoPoint;

inputPhotoEmpty#1cd7bf0d = InputPhoto;
inputPhoto#3bb3b94a id:long access_hash:long file_reference:bytes = InputPhoto;

inputFileLocation#dfdaabe1 volume_id:long local_id:int secret:long file_reference:bytes = InputFileLocation;
inputEncryptedFileLocation#f5235d55 id:long access_hash:long = InputFileLocation;
inputDocumentFileLocation#bad07584 id:long access_hash:long file_reference:bytes thumb_size:string = InputFileL
inputSecureFileLocation#cbc7ee28 id:long access_hash:long = InputFileLocation;
inputTakeoutFileLocation#29be5899 = InputFileLocation;
inputPhotoFileLocation#40181ffe id:long access_hash:long file_reference:bytes thumb_size:string = InputFileLoca
inputPeerPhotoFileLocation#27d69997 flags:# big:flags.0?true peer:InputPeer volume_id:long local_id:int = Input
inputStickerSetThumb#dbaeae9 stickerset:InputStickerSet volume_id:long local_id:int = InputFileLocation;

peerUser#9db1bc6d user_id:int = Peer;
peerChat#bad0e5bb chat_id:int = Peer;
peerChannel#bddde532 channel_id:int = Peer;

storage.fileUnknown#aa963b05 = storage.FileType;
storage.filePartial#40bc6f52 = storage.FileType;
storage.fileJpeg#7efe0e = storage.FileType;
storage.fileGif#cae1aadf = storage.FileType;
storage.filePng#a4f63c0 = storage.FileType;
storage.filePdf#ae1e508d = storage.FileType;
storage.fileMp3#528a0677 = storage.FileType;
storage.fileMov#4b09ebbc = storage.FileType;
storage.fileMp4#b3cea0e4 = storage.FileType;
storage.fileWebp#1081464c = storage.FileType;

userEmpty#200250ba id:int = User;
user#938458c1 flags:# self:flags.10?true contact:flags.11?true mutual_contact:flags.12?true deleted:flags.13?tr

userProfilePhotoEmpty#4f11bae1 = UserProfilePhoto;
userProfilePhoto#ecd75d8c photo_id:long photo_small:FileLocation photo_big:FileLocation dc_id:int = UserProfile

userStatusEmpty#9d05049 = UserStatus;
userStatusOnline#edb93949 expires:int = UserStatus;
userStatusOffline#8c703f was_online:int = UserStatus;
userStatusRecently#e26f42f1 = UserStatus;
userStatusLastWeek#7bf09fc = UserStatus;
userStatusLastMonth#77ebc742 = UserStatus;

chatEmpty#9ba2d800 id:int = Chat;
chat#3bda1bde flags:# creator:flags.0?true kicked:flags.1?true left:flags.2?true deactivated:flags.5?true id:in
chatForbidden#7328bdb id:int title:string = Chat;
channel#d31a961e flags:# creator:flags.0?true left:flags.2?true broadcast:flags.5?true verified:flags.7?true me
channelForbidden#289da732 flags:# broadcast:flags.5?true megagroup:flags.8?true id:int access_hash:long title:s

chatFull#1b7c9db3 flags:# can_set_username:flags.7?true has_scheduled:flags.8?true id:int about:string particip
channelFull#2d895c74 flags:# can_view_participants:flags.3?true can_set_username:flags.6?true can_set_stickers:

chatParticipant#c8d7493e user_id:int inviter_id:int date:int = ChatParticipant;
chatParticipantCreator#da13538a user_id:int = ChatParticipant;
chatParticipantAdmin#e2d6e436 user_id:int inviter_id:int date:int = ChatParticipant;

chatParticipantsForbidden#fc900c2b flags:# chat_id:int self_participant:flags.0?ChatParticipant = ChatParticipa
chatParticipants#3f460fed chat_id:int participants:Vector<ChatParticipant> version:int = ChatParticipants;

chatPhotoEmpty#37c1011c = ChatPhoto;
chatPhoto#475cdbd5 photo_small:FileLocation photo_big:FileLocation dc_id:int = ChatPhoto;
```

```
messageEmpty#83e5de54 id:int = Message;
message#452c0e65 flags:# out:flags.1?true mentioned:flags.4?true media_unread:flags.5?true silent:flags.13?true
messageService#9e19a1f6 flags:# out:flags.1?true mentioned:flags.4?true media_unread:flags.5?true silent:flags.

messageMediaEmpty#3ded6320 = MessageMedia;
messageMediaPhoto#695150d7 flags:# photo:flags.0?Photo ttl_seconds:flags.2?int = MessageMedia;
messageMediaGeo#56e0d474 geo:GeoPoint = MessageMedia;
messageMediaContact#cbf24940 phone_number:string first_name:string last_name:string vcard:string user_id:int =
messageMediaUnsupported#9f84f49e = MessageMedia;
messageMediaDocument#9cb070d7 flags:# document:flags.0?Document ttl_seconds:flags.2?int = MessageMedia;
messageMediaWebPage#a32dd600 webpage:WebPage = MessageMedia;
messageMediaVenue#2ec0533f geo:GeoPoint title:string address:string provider:string venue_id:string venue_type:
messageMediaGame#fdb19008 game:Game = MessageMedia;
messageMediaInvoice#84551347 flags:# shipping_address_requested:flags.1?true test:flags.3?true title:string des
messageMediaGeoLive#7c3c2609 geo:GeoPoint period:int = MessageMedia;
messageMediaPoll#4bd6e798 poll:Poll results:PollResults = MessageMedia;

messageActionEmpty#b6aef7b0 = MessageAction;
messageActionChatCreate#a6638b9a title:string users:Vector<int> = MessageAction;
messageActionChatEditTitle#b5a1ce5a title:string = MessageAction;
messageActionChatEditPhoto#7fcb13a8 photo:Photo = MessageAction;
messageActionChatDeletePhoto#95e3fbef = MessageAction;
messageActionChatAddUser#488a7337 users:Vector<int> = MessageAction;
messageActionChatDeleteUser#b2ae9b0c user_id:int = MessageAction;
messageActionChatJoinedByLink#f89cf5e8 inviter_id:int = MessageAction;
messageActionChannelCreate#95d2ac92 title:string = MessageAction;
messageActionChatMigrateTo#51bdb021 channel_id:int = MessageAction;
messageActionChannelMigrateFrom#b055eaee title:string chat_id:int = MessageAction;
messageActionPinMessage#94bd38ed = MessageAction;
messageActionHistoryClear#9fbab604 = MessageAction;
messageActionGameScore#92a72876 game_id:long score:int = MessageAction;
messageActionPaymentSentMe#8f31b327 flags:# currency:string total_amount:long payload:bytes info:flags.0?Paymen
messageActionPaymentSent#40699cd0 currency:string total_amount:long = MessageAction;
messageActionPhoneCall#80e11a7f flags:# video:flags.2?true call_id:long reason:flags.0?PhoneCallDiscardReason d
messageActionScreenshotTaken#4792929b = MessageAction;
messageActionCustomAction#fae69f56 message:string = MessageAction;
messageActionBotAllowed#abe9affe domain:string = MessageAction;
messageActionSecureValuesSentMe#1b287353 values:Vector<SecureValue> credentials:SecureCredentialsEncrypted = Me
messageActionSecureValuesSent#d95c6154 types:Vector<SecureValueType> = MessageAction;
messageActionContactSignUp#f3f25f76 = MessageAction;

dialog#2c171f72 flags:# pinned:flags.2?true unread_mark:flags.3?true peer:Peer top_message:int read_inbox_max_i
dialogFolder#71bd134c flags:# pinned:flags.2?true folder:Folder peer:Peer top_message:int unread_muted_peers_co

photoEmpty#2331b22d id:long = Photo;
photo#d07504a5 flags:# has_stickers:flags.0?true id:long access_hash:long file_reference:bytes date:int sizes:V

photoSizeEmpty#e17e23c type:string = PhotoSize;
photoSize#77bfb61b type:string location:FileLocation w:int h:int size:int = PhotoSize;
photoCachedSize#e9a734fa type:string location:FileLocation w:int h:int bytes:bytes = PhotoSize;
photoStrippedSize#e0b0bc2e type:string bytes:bytes = PhotoSize;

geoPointEmpty#1117dd5f = GeoPoint;
geoPoint#296f104 long:double lat:double access_hash:long = GeoPoint;

auth.sentCode#5e002502 flags:# type:auth.SentCodeType phone_code_hash:string next_type:flags.1?auth.CodeType ti

auth.authorization#cd050916 flags:# tmp_sessions:flags.0?int user:User = auth.Authorization;
auth.authorizationSignUpRequired#44747e9a flags:# terms_of_service:flags.0?help.TermsOfService = auth.Authoriza

auth.exportedAuthorization#df969c2d id:int bytes:bytes = auth.ExportedAuthorization;

inputNotifyPeer#b8bc5b0c peer:InputPeer = InputNotifyPeer;
inputNotifyUsers#193b4417 = InputNotifyPeer;
inputNotifyChats#4a95e84e = InputNotifyPeer;
inputNotifyBroadcasts#b1db7c7e = InputNotifyPeer;

inputPeerNotifySettings#9c3d198e flags:# show_previews:flags.0?Bool silent:flags.1?Bool mute_until:flags.2?int

peerNotifySettings#af509d20 flags:# show_previews:flags.0?Bool silent:flags.1?Bool mute_until:flags.2?int sound

peerSettings#818426cd flags:# report_spam:flags.0?true add_contact:flags.1?true block_contact:flags.2?true shar

wallPaper#a437c3ed id:long flags:# creator:flags.0?true default:flags.1?true pattern:flags.3?true dark:flags.4?

inputReportReasonSpam#58dbcab8 = ReportReason;
inputReportReasonViolence#1e22c78d = ReportReason;
inputReportReasonPornography#2e59d922 = ReportReason;
inputReportReasonChildAbuse#adf44ee3 = ReportReason;
inputReportReasonOther#e1746d0a text:string = ReportReason;
inputReportReasonCopyright#9b89f93a = ReportReason;
inputReportReasonGeoIrrelevant#dbd4feed = ReportReason;

userFull#edf17c12 flags:# blocked:flags.0?true phone_calls_available:flags.4?true phone_calls_private:flags.5?t

contact#f911c994 user_id:int mutual:Bool = Contact;

importedContact#d0028438 user_id:int client_id:long = ImportedContact;

contactBlocked#561bc879 user_id:int date:int = ContactBlocked;

contactStatus#d3680c61 user_id:int status:UserStatus = ContactStatus;

contacts.contactsNotModified#b74ba9d2 = contacts.Contacts;
contacts.contacts#eae87e42 contacts:Vector<Contact> saved_count:int users:Vector<User> = contacts.Contacts;

contacts.importedContacts#77d01c3b imported:Vector<ImportedContact> popular_invites:Vector<PopularContact> retr

contacts.blocked#1c138d15 blocked:Vector<ContactBlocked> users:Vector<User> = contacts.Blocked;
contacts.blockedSlice#900802a1 count:int blocked:Vector<ContactBlocked> users:Vector<User> = contacts.Blocked;

messages.dialogs#15ba6c40 dialogs:Vector<Dialog> messages:Vector<Message> chats:Vector<Chat> users:Vector<User>
messages.dialogsSlice#71e094f3 count:int dialogs:Vector<Dialog> messages:Vector<Message> chats:Vector<Chat> use
messages.dialogsNotModified#f0e3e596 count:int = messages.Dialogs;

messages.messages#8c718e87 messages:Vector<Message> chats:Vector<Chat> users:Vector<User> = messages.Messages;
messages.messagesSlice#c8edce1e flags:# inexact:flags.1?true count:int next_rate:flags.0?int messages:Vector<Me
messages.channelMessages#99262e37 flags:# inexact:flags.1?true pts:int count:int messages:Vector<Message> chats
messages.messagesNotModified#74535f21 count:int = messages.Messages;

messages.chats#64ff9fd5 chats:Vector<Chat> = messages.Chats;
messages.chatsSlice#9cd81144 count:int chats:Vector<Chat> = messages.Chats;

messages.chatFull#e5d7d19c full_chat:ChatFull chats:Vector<Chat> users:Vector<User> = messages.ChatFull;

messages.affectedHistory#b45c69d1 pts:int pts_count:int offset:int = messages.AffectedHistory;

inputMessagesFilterEmpty#57e2f66c = MessagesFilter;
inputMessagesFilterPhotos#9609a51c = MessagesFilter;
inputMessagesFilterVideo#9fc00e65 = MessagesFilter;
inputMessagesFilterPhotoVideo#56e9f0e4 = MessagesFilter;
inputMessagesFilterDocument#9eddf188 = MessagesFilter;
inputMessagesFilterUrl#7ef0dd87 = MessagesFilter;
inputMessagesFilterGif#ffc86587 = MessagesFilter;
inputMessagesFilterVoice#50f5c392 = MessagesFilter;
inputMessagesFilterMusic#3751b49e = MessagesFilter;
inputMessagesFilterChatPhotos#3a20ecb8 = MessagesFilter;
inputMessagesFilterPhoneCalls#80c99768 flags:# missed:flags.0?true = MessagesFilter;
inputMessagesFilterRoundVoice#7a7c17a4 = MessagesFilter;
inputMessagesFilterRoundVideo#b549da53 = MessagesFilter;
inputMessagesFilterMyMentions#c1f8e69a = MessagesFilter;
```

```
inputMessagesFilterGeo#e7026d0d = MessagesFilter;
inputMessagesFilterContact#e062db83 = MessagesFilter;

updateNewMessage#1f2b0afd message:Message pts:int pts_count:int = Update;
updateMessageID#4e90bfd6 id:int random_id:long = Update;
updateDeleteMessages#a20db0e5 messages:Vector<int> pts:int pts_count:int = Update;
updateUserTyping#5c486927 user_id:int action:SendMessageAction = Update;
updateChatUserTyping#9a65ea1f chat_id:int user_id:int action:SendMessageAction = Update;
updateChatParticipants#7761198 participants:ChatParticipants = Update;
updateUserStatus#1bfbd823 user_id:int status:UserStatus = Update;
updateUserName#a7332b73 user_id:int first_name:string last_name:string username:string = Update;
updateUserPhoto#95313b0c user_id:int date:int photo:UserProfilePhoto previous:Bool = Update;
updateNewEncryptedMessage#12bcbd9a message:EncryptedMessage qts:int = Update;
updateEncryptedChatTyping#1710f156 chat_id:int = Update;
updateEncryption#b4a2e88d chat:EncryptedChat date:int = Update;
updateEncryptedMessagesRead#38fe25b7 chat_id:int max_date:int date:int = Update;
updateChatParticipantAdd#ea4b0e5c chat_id:int user_id:int inviter_id:int date:int version:int = Update;
updateChatParticipantDelete#6e5f8c22 chat_id:int user_id:int version:int = Update;
updateDcOptions#8e5e9873 dc_options:Vector<DcOption> = Update;
updateUserBlocked#80ece81a user_id:int blocked:Bool = Update;
updateNotifySettings#bec268ef peer:NotifyPeer notify_settings:PeerNotifySettings = Update;
updateServiceNotification#ebe46819 flags:# popup:flags.0?true inbox_date:flags.1?int type:string message:string
updatePrivacy#ee3b272a key:PrivacyKey rules:Vector<PrivacyRule> = Update;
updateUserPhone#12b9417b user_id:int phone:string = Update;
updateReadHistoryInbox#9c974fdf flags:# folder_id:flags.0?int peer:Peer max_id:int still_unread_count:int pts:i
updateReadHistoryOutbox#2f2f21bf peer:Peer max_id:int pts:int pts_count:int = Update;
updateWebPage#7f891213 webpage:WebPage pts:int pts_count:int = Update;
updateReadMessagesContents#68c13933 messages:Vector<int> pts:int pts_count:int = Update;
updateChannelTooLong#eb0467fb flags:# channel_id:int pts:flags.0?int = Update;
updateChannel#b6d45656 channel_id:int = Update;
updateNewChannelMessage#62ba04d9 message:Message pts:int pts_count:int = Update;
updateReadChannelInbox#330b5424 flags:# folder_id:flags.0?int channel_id:int max_id:int still_unread_count:int
updateDeleteChannelMessages#c37521c9 channel_id:int messages:Vector<int> pts:int pts_count:int = Update;
updateChannelMessageViews#98a12b4b channel_id:int id:int views:int = Update;
updateChatParticipantAdmin#b6901959 chat_id:int user_id:int is_admin:Bool version:int = Update;
updateNewStickerSet#688a30aa stickerset:messages.StickerSet = Update;
updateStickerSetsOrder#bb2d201 flags:# masks:flags.0?true order:Vector<long> = Update;
updateStickerSets#43ae3dec = Update;
updateSavedGifs#9375341e = Update;
updateBotInlineQuery#54826690 flags:# query_id:long user_id:int query:string geo:flags.0?GeoPoint offset:string
updateBotInlineSend#e48f964 flags:# user_id:int query:string geo:flags.0?GeoPoint id:string msg_id:flags.1?Inpu
updateEditChannelMessage#1b3f4df7 message:Message pts:int pts_count:int = Update;
updateChannelPinnedMessage#98592475 channel_id:int id:int = Update;
updateBotCallbackQuery#e73547e1 flags:# query_id:long user_id:int peer:Peer msg_id:int chat_instance:long data:
updateEditMessage#e40370a3 message:Message pts:int pts_count:int = Update;
updateInlineBotCallbackQuery#f9d27a5a flags:# query_id:long user_id:int msg_id:InputBotInlineMessageID chat_ins
updateReadChannelOutbox#25d6c9c7 channel_id:int max_id:int = Update;
updateDraftMessage#ee2bb969 peer:Peer draft:DraftMessage = Update;
updateReadFeaturedStickers#571d2742 = Update;
updateRecentStickers#9a422c20 = Update;
updateConfig#a229dd06 = Update;
updatePtsChanged#3354678f = Update;
updateChannelWebPage#40771900 channel_id:int webpage:WebPage pts:int pts_count:int = Update;
updateDialogPinned#6e6fe51c flags:# pinned:flags.0?true folder_id:flags.1?int peer:DialogPeer = Update;
updatePinnedDialogs#fa0f3ca2 flags:# folder_id:flags.1?int order:flags.0?Vector<DialogPeer> = Update;
updateBotWebhookJSON#8317c0c3 data:DataJSON = Update;
updateBotWebhookJSONQuery#9b9240a6 query_id:long data:DataJSON timeout:int = Update;
updateBotShippingQuery#e0cdc940 query_id:long user_id:int payload:bytes shipping_address:PostAddress = Update;
updateBotPrecheckoutQuery#5d2f3aa9 flags:# query_id:long user_id:int payload:bytes info:flags.0?PaymentRequeste
updatePhoneCall#ab0f6b1e phone_call:PhoneCall = Update;
updateLangPackTooLong#46560264 lang_code:string = Update;
updateLangPack#56022f4d difference:LangPackDifference = Update;
updateFavedStickers#e511996d = Update;
updateChannelReadMessagesContents#89893b45 channel_id:int messages:Vector<int> = Update;
updateContactsReset#7084a7be = Update;
updateChannelAvailableMessages#70db6837 channel_id:int available_min_id:int = Update;
updateDialogUnreadMark#e16459c3 flags:# unread:flags.0?true peer:DialogPeer = Update;
updateUserPinnedMessage#4c43da18 user_id:int id:int = Update;
updateChatPinnedMessage#e10db349 chat_id:int id:int version:int = Update;
updateMessagePoll#aca1657b flags:# poll_id:long poll:flags.0?Poll results:PollResults = Update;
updateChatDefaultBannedRights#54c01850 peer:Peer default_banned_rights:ChatBannedRights version:int = Update;
updateFolderPeers#19360dc0 folder_peers:Vector<FolderPeer> pts:int pts_count:int = Update;
updatePeerSettings#6a7e7366 peer:Peer settings:PeerSettings = Update;
updatePeerLocated#b4afcfb0 peers:Vector<PeerLocated> = Update;
updateNewScheduledMessage#39a51dfb message:Message = Update;
updateDeleteScheduledMessages#90866cee peer:Peer messages:Vector<int> = Update;
updateTheme#8216fba3 theme:Theme = Update;

updates.state#a56c2a3e pts:int qts:int date:int seq:int unread_count:int = updates.State;

updates.differenceEmpty#5d75a138 date:int seq:int = updates.Difference;
updates.difference#f49ca0 new_messages:Vector<Message> new_encrypted_messages:Vector<EncryptedMessage> other_up
updates.differenceSlice#a8fb1981 new_messages:Vector<Message> new_encrypted_messages:Vector<EncryptedMessage> o
updates.differenceTooLong#4afe8f6d pts:int = updates.Difference;

updatesTooLong#e317af7e = Updates;
updateShortMessage#914fbf11 flags:# out:flags.1?true mentioned:flags.4?true media_unread:flags.5?true silent:fl
updateShortChatMessage#16812688 flags:# out:flags.1?true mentioned:flags.4?true media_unread:flags.5?true silen
updateShort#78d4dec1 update:Update date:int = Updates;
updatesCombined#725b04c3 updates:Vector<Update> users:Vector<User> chats:Vector<Chat> date:int seq_start:int se
updates#74ae4240 updates:Vector<Update> users:Vector<User> chats:Vector<Chat> date:int seq:int = Updates;
updateShortSentMessage#11f1331c flags:# out:flags.1?true id:int pts:int pts_count:int date:int media:flags.9?Me

photos.photos#8dca6aa5 photos:Vector<Photo> users:Vector<User> = photos.Photos;
photos.photosSlice#15051f54 count:int photos:Vector<Photo> users:Vector<User> = photos.Photos;

photos.photo#20212ca8 photo:Photo users:Vector<User> = photos.Photo;

upload.file#96a18d5 type:storage.FileType mtime:int bytes:bytes = upload.File;
upload.fileCdnRedirect#f18cda44 dc_id:int file_token:bytes encryption_key:bytes encryption_iv:bytes file_hashes

dcOption#18b7a10d flags:# ipv6:flags.0?true media_only:flags.1?true tcpo_only:flags.2?true cdn:flags.3?true sta

config#330b4067 flags:# phonecalls_enabled:flags.1?true default_p2p_contacts:flags.3?true preload_featured_stic

nearestDc#8e1a1775 country:string this_dc:int nearest_dc:int = NearestDc;

help.appUpdate#1da7158f flags:# can_not_skip:flags.0?true id:int version:string text:string entities:Vector<Mes
help.noAppUpdate#c45a6536 = help.AppUpdate;

help.inviteText#18cb9f78 message:string = help.InviteText;

encryptedChatEmpty#ab7ec0a0 id:int = EncryptedChat;
encryptedChatWaiting#3bf703dc id:int access_hash:long date:int admin_id:int participant_id:int = EncryptedChat;
encryptedChatRequested#c878527e id:int access_hash:long date:int admin_id:int participant_id:int g_a:bytes = En
encryptedChat#fa56ce36 id:int access_hash:long date:int admin_id:int participant_id:int g_a_or_b:bytes key_fing
encryptedChatDiscarded#13d6dd27 id:int = EncryptedChat;

inputEncryptedChat#f141b5e1 chat_id:int access_hash:long = InputEncryptedChat;

encryptedFileEmpty#c21f497e = EncryptedFile;
encryptedFile#4a70994c id:long access_hash:long size:int dc_id:int key_fingerprint:int = EncryptedFile;

inputEncryptedFileEmpty#1837c364 = InputEncryptedFile;
inputEncryptedFileUploaded#64bd0306 id:long parts:int md5_checksum:string key_fingerprint:int = InputEncryptedF
inputEncryptedFile#5a17b5e5 id:long access_hash:long = InputEncryptedFile;
inputEncryptedFileBigUploaded#2dc173c8 id:long parts:int key_fingerprint:int = InputEncryptedFile;

encryptedMessage#ed18c118 random_id:long chat_id:int date:int bytes:bytes file:EncryptedFile = EncryptedMessage
encryptedMessageService#23734b06 random_id:long chat_id:int date:int bytes:bytes = EncryptedMessage;

messages.dhConfigNotModified#c0e24635 random:bytes = messages.DhConfig;
```

```
messages.dhConfig#2c221edd g:int p:bytes version:int random:bytes = messages.DhConfig;
messages.sentEncryptedMessage#560f8935 date:int = messages.SentEncryptedMessage;
messages.sentEncryptedFile#9493ff32 date:int file:EncryptedFile = messages.SentEncryptedMessage;

inputDocumentEmpty#72f0eaae = InputDocument;
inputDocument#1abfb575 id:long access_hash:long file_reference:bytes = InputDocument;

documentEmpty#36f8c871 id:long = Document;
document#9ba29cc1 flags:# id:long access_hash:long file_reference:bytes date:int mime_type:string size:int thum

help.support#17c6b5f6 phone_number:string user:User = help.Support;

notifyPeer#9fd40bd8 peer:Peer = NotifyPeer;
notifyUsers#b4c83b4c = NotifyPeer;
notifyChats#c007cec3 = NotifyPeer;
notifyBroadcasts#d612e8ef = NotifyPeer;

sendMessageTypingAction#16bf744e = SendMessageAction;
sendMessageCancelAction#fd5ec8f5 = SendMessageAction;
sendMessageRecordVideoAction#a187d66f = SendMessageAction;
sendMessageUploadVideoAction#e9763aec progress:int = SendMessageAction;
sendMessageRecordAudioAction#d52f73f7 = SendMessageAction;
sendMessageUploadAudioAction#f351d7ab progress:int = SendMessageAction;
sendMessageUploadPhotoAction#d1d34a26 progress:int = SendMessageAction;
sendMessageUploadDocumentAction#aa0cd9e4 progress:int = SendMessageAction;
sendMessageGeoLocationAction#176f8ba1 = SendMessageAction;
sendMessageChooseContactAction#628cbc6f = SendMessageAction;
sendMessageGamePlayAction#dd6a8f48 = SendMessageAction;
sendMessageRecordRoundAction#88f27fbc = SendMessageAction;
sendMessageUploadRoundAction#243e1c66 progress:int = SendMessageAction;

contacts.found#b3134d9d my_results:Vector<Peer> results:Vector<Peer> chats:Vector<Chat> users:Vector<User> = co

inputPrivacyKeyStatusTimestamp#4f96cb18 = InputPrivacyKey;
inputPrivacyKeyChatInvite#bdfb0426 = InputPrivacyKey;
inputPrivacyKeyPhoneCall#fabadc5f = InputPrivacyKey;
inputPrivacyKeyPhoneP2P#db9e70d2 = InputPrivacyKey;
inputPrivacyKeyForwards#a4dd4c08 = InputPrivacyKey;
inputPrivacyKeyProfilePhoto#5719bacc = InputPrivacyKey;
inputPrivacyKeyPhoneNumber#352dafa = InputPrivacyKey;
inputPrivacyKeyAddedByPhone#d1219bdd = InputPrivacyKey;

privacyKeyStatusTimestamp#bc2eab30 = PrivacyKey;
privacyKeyChatInvite#500e6dfa = PrivacyKey;
privacyKeyPhoneCall#3d662b7b = PrivacyKey;
privacyKeyPhoneP2P#39491cc8 = PrivacyKey;
privacyKeyForwards#69ec56a3 = PrivacyKey;
privacyKeyProfilePhoto#96151fed = PrivacyKey;
privacyKeyPhoneNumber#d19ae46d = PrivacyKey;
privacyKeyAddedByPhone#42ffd42b = PrivacyKey;

inputPrivacyValueAllowContacts#d09e07b = InputPrivacyRule;
inputPrivacyValueAllowAll#184b35ce = InputPrivacyRule;
inputPrivacyValueAllowUsers#131cc67f users:Vector<InputUser> = InputPrivacyRule;
inputPrivacyValueDisallowContacts#ba52007 = InputPrivacyRule;
inputPrivacyValueDisallowAll#d66b66c9 = InputPrivacyRule;
inputPrivacyValueDisallowUsers#90110467 users:Vector<InputUser> = InputPrivacyRule;
inputPrivacyValueAllowChatParticipants#4c81c1ba chats:Vector<int> = InputPrivacyRule;
inputPrivacyValueDisallowChatParticipants#d82363af chats:Vector<int> = InputPrivacyRule;

privacyValueAllowContacts#fffe1bac = PrivacyRule;
privacyValueAllowAll#65427b82 = PrivacyRule;
privacyValueAllowUsers#4d5bbe0c users:Vector<int> = PrivacyRule;
privacyValueDisallowContacts#f888fa1a = PrivacyRule;
privacyValueDisallowAll#8b73e763 = PrivacyRule;
privacyValueDisallowUsers#c7f49b7 users:Vector<int> = PrivacyRule;
privacyValueAllowChatParticipants#18be796b chats:Vector<int> = PrivacyRule;
privacyValueDisallowChatParticipants#acae0690 chats:Vector<int> = PrivacyRule;

account.privacyRules#50a04e45 rules:Vector<PrivacyRule> chats:Vector<Chat> users:Vector<User> = account.Privacy

accountDaysTTL#b8d0afdf days:int = AccountDaysTTL;

documentAttributeImageSize#6c37c15c w:int h:int = DocumentAttribute;
documentAttributeAnimated#11b58939 = DocumentAttribute;
documentAttributeSticker#6319d612 flags:# mask:flags.1?true alt:string stickerset:InputStickerSet mask_coords:f
documentAttributeVideo#ef02ce6 flags:# round_message:flags.0?true supports_streaming:flags.1?true duration:int
documentAttributeAudio#9852f9c6 flags:# voice:flags.10?true duration:int title:flags.0?string performer:flags.1
documentAttributeFilename#15590068 file_name:string = DocumentAttribute;
documentAttributeHasStickers#9801d2f7 = DocumentAttribute;

messages.stickersNotModified#f1749a22 = messages.Stickers;
messages.stickers#e4599bbd hash:int stickers:Vector<Document> = messages.Stickers;

stickerPack#12b299d4 emoticon:string documents:Vector<long> = StickerPack;

messages.allStickersNotModified#e86602c3 = messages.AllStickers;
messages.allStickers#edfd405f hash:int sets:Vector<StickerSet> = messages.AllStickers;

messages.affectedMessages#84d19185 pts:int pts_count:int = messages.AffectedMessages;

webPageEmpty#eb1477e8 id:long = WebPage;
webPagePending#c586da1c id:long date:int = WebPage;
webPage#fa64e172 flags:# id:long url:string display_url:string hash:int type:flags.0?string site_name:flags.1?s
webPageNotModified#85849473 = WebPage;

authorization#ad01d61d flags:# current:flags.0?true official_app:flags.1?true password_pending:flags.2?true has

account.authorizations#1250abde authorizations:Vector<Authorization> = account.Authorizations;

account.password#ad2641f8 flags:# has_recovery:flags.0?true has_secure_values:flags.1?true has_password:flags.2

account.passwordSettings#9a5c33e5 flags:# email:flags.0?string secure_settings:flags.1?SecureSecretSettings = a

account.passwordInputSettings#c23727c9 flags:# new_algo:flags.0?PasswordKdfAlgo new_password_hash:flags.0?bytes

auth.passwordRecovery#137948a5 email_pattern:string = auth.PasswordRecovery;

receivedNotifyMessage#a384b779 id:int flags:int = ReceivedNotifyMessage;

chatInviteEmpty#69df3769 = ExportedChatInvite;
chatInviteExported#fc2e05bc link:string = ExportedChatInvite;

chatInviteAlready#5a686d7c chat:Chat = ChatInvite;
chatInvite#dfc2f58e flags:# channel:flags.0?true broadcast:flags.1?true public:flags.2?true megagroup:flags.3?t

inputStickerSetEmpty#ffb62b95 = InputStickerSet;
inputStickerSetID#9de7a269 id:long access_hash:long = InputStickerSet;
inputStickerSetShortName#861cc8a0 short_name:string = InputStickerSet;
inputStickerSetAnimatedEmoji#28703c8 = InputStickerSet;

stickerSet#eeb46f27 flags:# archived:flags.1?true official:flags.2?true masks:flags.3?true animated:flags.5?tru

messages.stickerSet#b60a24a6 set:StickerSet packs:Vector<StickerPack> documents:Vector<Document> = messages.Sti

botCommand#c27ac8c7 command:string description:string = BotCommand;

botInfo#98e81d3a user_id:int description:string commands:Vector<BotCommand> = BotInfo;

keyboardButton#a2fa4880 text:string = KeyboardButton;
```

```
keyboardButtonUrl#258aff05 text:string url:string = KeyboardButton;
keyboardButtonCallback#683a5e46 text:string data:bytes = KeyboardButton;
keyboardButtonRequestPhone#b16a6c29 text:string = KeyboardButton;
keyboardButtonRequestGeoLocation#fc796b3f text:string = KeyboardButton;
keyboardButtonSwitchInline#568a748 flags:# same_peer:flags.0?true text:string query:string = KeyboardButton;
keyboardButtonGame#50f41ccf text:string = KeyboardButton;
keyboardButtonBuy#afd93fbb text:string = KeyboardButton;
keyboardButtonUrlAuth#10b78d29 flags:# text:string fwd_text:flags.0?string url:string button_id:int = KeyboardB
inputKeyboardButtonUrlAuth#d02e7fd4 flags:# request_write_access:flags.0?true text:string fwd_text:flags.1?stri

keyboardButtonRow#77608b83 buttons:Vector<KeyboardButton> = KeyboardButtonRow;

replyKeyboardHide#a03e5b85 flags:# selective:flags.2?true = ReplyMarkup;
replyKeyboardForceReply#f4108aa0 flags:# single_use:flags.1?true selective:flags.2?true = ReplyMarkup;
replyKeyboardMarkup#3502758c flags:# resize:flags.0?true single_use:flags.1?true selective:flags.2?true rows:Ve
replyInlineMarkup#48a30254 rows:Vector<KeyboardButtonRow> = ReplyMarkup;

messageEntityUnknown#bb92ba95 offset:int length:int = MessageEntity;
messageEntityMention#fa04579d offset:int length:int = MessageEntity;
messageEntityHashtag#6f635b0d offset:int length:int = MessageEntity;
messageEntityBotCommand#6cef8ac7 offset:int length:int = MessageEntity;
messageEntityUrl#6ed02538 offset:int length:int = MessageEntity;
messageEntityEmail#64e475c2 offset:int length:int = MessageEntity;
messageEntityBold#bd610bc9 offset:int length:int = MessageEntity;
messageEntityItalic#826f8b60 offset:int length:int = MessageEntity;
messageEntityCode#28a20571 offset:int length:int = MessageEntity;
messageEntityPre#73924be0 offset:int length:int language:string = MessageEntity;
messageEntityTextUrl#76a6d327 offset:int length:int url:string = MessageEntity;
messageEntityMentionName#352dca58 offset:int length:int user_id:int = MessageEntity;
inputMessageEntityMentionName#208e68c9 offset:int length:int user_id:InputUser = MessageEntity;
messageEntityPhone#9b69e34b offset:int length:int = MessageEntity;
messageEntityCashtag#4c4e743f offset:int length:int = MessageEntity;
messageEntityUnderline#9c4e7e8b offset:int length:int = MessageEntity;
messageEntityStrike#bf0693d4 offset:int length:int = MessageEntity;
messageEntityBlockquote#20df5d0 offset:int length:int = MessageEntity;

inputChannelEmpty#ee8c1e86 = InputChannel;
inputChannel#afeb712e channel_id:int access_hash:long = InputChannel;
inputChannelFromMessage#2a286531 peer:InputPeer msg_id:int channel_id:int = InputChannel;

contacts.resolvedPeer#7f077ad9 peer:Peer chats:Vector<Chat> users:Vector<User> = contacts.ResolvedPeer;

messageRange#ae30253 min_id:int max_id:int = MessageRange;

updates.channelDifferenceEmpty#3e11affb flags:# final:flags.0?true pts:int timeout:flags.1?int = updates.Channe
updates.channelDifferenceTooLong#a4bcc6fe flags:# final:flags.0?true timeout:flags.1?int dialog:Dialog messages
updates.channelDifference#2064674e flags:# final:flags.0?true pts:int timeout:flags.1?int new_messages:Vector<M

channelMessagesFilterEmpty#94d42ee7 = ChannelMessagesFilter;
channelMessagesFilter#cd77d957 flags:# exclude_new_messages:flags.1?true ranges:Vector<MessageRange> = ChannelM

channelParticipant#15ebac1d user_id:int date:int = ChannelParticipant;
channelParticipantSelf#a3289a6d user_id:int inviter_id:int date:int = ChannelParticipant;
channelParticipantCreator#808d15a4 flags:# user_id:int rank:flags.0?string = ChannelParticipant;
channelParticipantAdmin#ccbebbaf flags:# can_edit:flags.0?true self:flags.1?true user_id:int inviter_id:flags.1
channelParticipantBanned#1c0facaf flags:# left:flags.0?true user_id:int kicked_by:int date:int banned_rights:Ch

channelParticipantsRecent#de3f3c79 = ChannelParticipantsFilter;
channelParticipantsAdmins#b4608969 = ChannelParticipantsFilter;
channelParticipantsKicked#a3b54985 q:string = ChannelParticipantsFilter;
channelParticipantsBots#b0d1865b = ChannelParticipantsFilter;
channelParticipantsBanned#1427a5e1 q:string = ChannelParticipantsFilter;
channelParticipantsSearch#656ac4b q:string = ChannelParticipantsFilter;
channelParticipantsContacts#bb6ae88d q:string = ChannelParticipantsFilter;

channels.channelParticipants#f56ee2a8 count:int participants:Vector<ChannelParticipant> users:Vector<User> = ch
channels.channelParticipantsNotModified#f0173fe9 = channels.ChannelParticipants;

channels.channelParticipant#d0d9b163 participant:ChannelParticipant users:Vector<User> = channels.ChannelPartic

help.termsOfService#780a0310 flags:# popup:flags.0?true id:DataJSON text:string entities:Vector<MessageEntity>

foundGif#162ecc1f url:string thumb_url:string content_url:string content_type:string w:int h:int = FoundGif;
foundGifCached#9c750409 url:string photo:Photo document:Document = FoundGif;

messages.foundGifs#450a1c0a next_offset:int results:Vector<FoundGif> = messages.FoundGifs;

messages.savedGifsNotModified#e8025ca2 = messages.SavedGifs;
messages.savedGifs#2e0709a5 hash:int gifs:Vector<Document> = messages.SavedGifs;

inputBotInlineMessageMediaAuto#3380c786 flags:# message:string entities:flags.1?Vector<MessageEntity> reply_mar
inputBotInlineMessageText#3dcd7a87 flags:# no_webpage:flags.0?true message:string entities:flags.1?Vector<Messa
inputBotInlineMessageMediaGeo#c1b15d65 flags:# geo_point:InputGeoPoint period:int reply_markup:flags.2?ReplyMar
inputBotInlineMessageMediaVenue#417bbf11 flags:# geo_point:InputGeoPoint title:string address:string provider:s
inputBotInlineMessageMediaContact#a6edbffd flags:# phone_number:string first_name:string last_name:string vcard
inputBotInlineMessageGame#4b425864 flags:# reply_markup:flags.2?ReplyMarkup = InputBotInlineMessage;

inputBotInlineResult#88bf9319 flags:# id:string type:string title:flags.1?string description:flags.2?string url
inputBotInlineResultPhoto#a8d864a7 id:string type:string photo:InputPhoto send_message:InputBotInlineMessage =
inputBotInlineResultDocument#fff8fdc4 flags:# id:string type:string title:flags.1?string description:flags.2?st
inputBotInlineResultGame#4fa417f2 id:string short_name:string send_message:InputBotInlineMessage = InputBotInli

botInlineMessageMediaAuto#764cf810 flags:# message:string entities:flags.1?Vector<MessageEntity> reply_markup:f
botInlineMessageText#8c7f65e2 flags:# no_webpage:flags.0?true message:string entities:flags.1?Vector<MessageEnt
botInlineMessageMediaGeo#b722de65 flags:# geo:GeoPoint period:int reply_markup:flags.2?ReplyMarkup = BotInlineM
botInlineMessageMediaVenue#8a86659c flags:# geo:GeoPoint title:string address:string provider:string venue_id:s
botInlineMessageMediaContact#18d1cdc2 flags:# phone_number:string first_name:string last_name:string vcard:stri

botInlineResult#11965f3a flags:# id:string type:string title:flags.1?string description:flags.2?string url:flag
botInlineMediaResult#17db940b flags:# id:string type:string photo:flags.0?Photo document:flags.1?Document title

messages.botResults#947ca848 flags:# gallery:flags.0?true query_id:long next_offset:flags.1?string switch_pm:fl

exportedMessageLink#5dab1af4 link:string html:string = ExportedMessageLink;

messageFwdHeader#ec338270 flags:# from_id:flags.0?int from_name:flags.5?string date:int channel_id:flags.1?int

auth.codeTypeSms#72a3158c = auth.CodeType;
auth.codeTypeCall#741cd3e3 = auth.CodeType;
auth.codeTypeFlashCall#226ccefb = auth.CodeType;

auth.sentCodeTypeApp#3dbb5986 length:int = auth.SentCodeType;
auth.sentCodeTypeSms#c000bba2 length:int = auth.SentCodeType;
auth.sentCodeTypeCall#5353e5a7 length:int = auth.SentCodeType;
auth.sentCodeTypeFlashCall#ab03c6d9 pattern:string = auth.SentCodeType;

messages.botCallbackAnswer#36585ea4 flags:# alert:flags.1?true has_url:flags.3?true native_ui:flags.4?true mess

messages.messageEditData#26b5dde6 flags:# caption:flags.0?true = messages.MessageEditData;

inputBotInlineMessageID#890c3d89 dc_id:int id:long access_hash:long = InputBotInlineMessageID;

inlineBotSwitchPM#3c20629f text:string start_param:string = InlineBotSwitchPM;

messages.peerDialogs#3371c354 dialogs:Vector<Dialog> messages:Vector<Message> chats:Vector<Chat> users:Vector<U

topPeer#edcdc05b peer:Peer rating:double = TopPeer;

topPeerCategoryBotsPM#ab661b5b = TopPeerCategory;
topPeerCategoryBotsInline#148677e2 = TopPeerCategory;
topPeerCategoryCorrespondents#637b7ed = TopPeerCategory;
topPeerCategoryGroups#bd17a14a = TopPeerCategory;
```

```
topPeerCategoryChannels#161d9628 = TopPeerCategory;
topPeerCategoryPhoneCalls#1e76c65e = TopPeerCategory;
topPeerCategoryForwardUsers#a8406ca9 = TopPeerCategory;
topPeerCategoryForwardChats#fbeec0f0 = TopPeerCategory;

topPeerCategoryPeers#fb834291 category:TopPeerCategory count:int peers:Vector<TopPeer> = TopPeerCategoryPeers;

contacts.topPeersNotModified#de266ef5 = contacts.TopPeers;
contacts.topPeers#70b772a8 categories:Vector<TopPeerCategoryPeers> chats:Vector<Chat> users:Vector<User> = cont
contacts.topPeersDisabled#b52c939d = contacts.TopPeers;

draftMessageEmpty#1b0c841a flags:# date:flags.0?int = DraftMessage;
draftMessage#fd8e711f flags:# no_webpage:flags.1?true reply_to_msg_id:flags.0?int message:string entities:flags

messages.featuredStickersNotModified#4ede3cf = messages.FeaturedStickers;
messages.featuredStickers#f89d88e5 hash:int sets:Vector<StickerSetCovered> unread:Vector<long> = messages.Featu

messages.recentStickersNotModified#b17f890 = messages.RecentStickers;
messages.recentStickers#22f3afb3 hash:int packs:Vector<StickerPack> stickers:Vector<Document> dates:Vector<int>

messages.archivedStickers#4fcba9c8 count:int sets:Vector<StickerSetCovered> = messages.ArchivedStickers;

messages.stickerSetInstallResultSuccess#38641628 = messages.StickerSetInstallResult;
messages.stickerSetInstallResultArchive#35e410a8 sets:Vector<StickerSetCovered> = messages.StickerSetInstallRes

stickerSetCovered#6410a5d2 set:StickerSet cover:Document = StickerSetCovered;
stickerSetMultiCovered#3407e51b set:StickerSet covers:Vector<Document> = StickerSetCovered;

maskCoords#aed6dbb2 n:int x:double y:double zoom:double = MaskCoords;

inputStickeredMediaPhoto#4a992157 id:InputPhoto = InputStickeredMedia;
inputStickeredMediaDocument#438865b id:InputDocument = InputStickeredMedia;

game#bdf9653b flags:# id:long access_hash:long short_name:string title:string description:string photo:Photo do

inputGameID#32c3e77 id:long access_hash:long = InputGame;
inputGameShortName#c331e80a bot_id:InputUser short_name:string = InputGame;

highScore#58fffcd0 pos:int user_id:int score:int = HighScore;

messages.highScores#9a3bfd99 scores:Vector<HighScore> users:Vector<User> = messages.HighScores;

textEmpty#dc3d824f = RichText;
textPlain#744694e0 text:string = RichText;
textBold#6724abc4 text:RichText = RichText;
textItalic#d912a59c text:RichText = RichText;
textUnderline#c12622c4 text:RichText = RichText;
textStrike#9bf8bb95 text:RichText = RichText;
textFixed#6c3f19b9 text:RichText = RichText;
textUrl#3c2884c1 text:RichText url:string webpage_id:long = RichText;
textEmail#de5a0dd6 text:RichText email:string = RichText;
textConcat#7e6260d7 texts:Vector<RichText> = RichText;
textSubscript#ed6a8504 text:RichText = RichText;
textSuperscript#c7fb5e01 text:RichText = RichText;
textMarked#34b8621 text:RichText = RichText;
textPhone#1ccb966a text:RichText phone:string = RichText;
textImage#81ccf4f document_id:long w:int h:int = RichText;
textAnchor#35553762 text:RichText name:string = RichText;

pageBlockUnsupported#13567e8a = PageBlock;
pageBlockTitle#70abc3fd text:RichText = PageBlock;
pageBlockSubtitle#8ffa9a1f text:RichText = PageBlock;
pageBlockAuthorDate#baafe5e0 author:RichText published_date:int = PageBlock;
pageBlockHeader#bfd064ec text:RichText = PageBlock;
pageBlockSubheader#f12bb6e1 text:RichText = PageBlock;
pageBlockParagraph#467a0766 text:RichText = PageBlock;
pageBlockPreformatted#c070d93e text:RichText language:string = PageBlock;
pageBlockFooter#48870999 text:RichText = PageBlock;
pageBlockDivider#db20b188 = PageBlock;
pageBlockAnchor#ce0d37b0 name:string = PageBlock;
pageBlockList#e4e88011 items:Vector<PageListItem> = PageBlock;
pageBlockBlockquote#263d7c26 text:RichText caption:RichText = PageBlock;
pageBlockPullquote#4f4456d3 text:RichText caption:RichText = PageBlock;
pageBlockPhoto#1759c560 flags:# photo_id:long caption:PageCaption url:flags.0?string webpage_id:flags.0?long =
pageBlockVideo#7c8fe7b6 flags:# autoplay:flags.0?true loop:flags.1?true video_id:long caption:PageCaption = Pag
pageBlockCover#39f23300 cover:PageBlock = PageBlock;
pageBlockEmbed#a8718dc5 flags:# full_width:flags.0?true allow_scrolling:flags.3?true url:flags.1?string html:fl
pageBlockEmbedPost#f259a80b url:string webpage_id:long author_photo_id:long author:string date:int blocks:Vecto
pageBlockCollage#65a0fa4d items:Vector<PageBlock> caption:PageCaption = PageBlock;
pageBlockSlideshow#31f9590 items:Vector<PageBlock> caption:PageCaption = PageBlock;
pageBlockChannel#ef1751b5 channel:Chat = PageBlock;
pageBlockAudio#804361ea audio_id:long caption:PageCaption = PageBlock;
pageBlockKicker#1e148390 text:RichText = PageBlock;
pageBlockTable#bf4dea82 flags:# bordered:flags.0?true striped:flags.1?true title:RichText rows:Vector<PageTable
pageBlockOrderedList#9a8ae1e1 items:Vector<PageListOrderedItem> = PageBlock;
pageBlockDetails#76768bed flags:# open:flags.0?true blocks:Vector<PageBlock> title:RichText = PageBlock;
pageBlockRelatedArticles#16115a96 title:RichText articles:Vector<PageRelatedArticle> = PageBlock;
pageBlockMap#a44f3ef6 geo:GeoPoint zoom:int w:int h:int caption:PageCaption = PageBlock;

phoneCallDiscardReasonMissed#85e42301 = PhoneCallDiscardReason;
phoneCallDiscardReasonDisconnect#e095c1a0 = PhoneCallDiscardReason;
phoneCallDiscardReasonHangup#57adc690 = PhoneCallDiscardReason;
phoneCallDiscardReasonBusy#faf7e8c9 = PhoneCallDiscardReason;

dataJSON#7d748d04 data:string = DataJSON;

labeledPrice#cb296bf8 label:string amount:long = LabeledPrice;

invoice#c30aa358 flags:# test:flags.0?true name_requested:flags.1?true phone_requested:flags.2?true email_reque

paymentCharge#ea02c27e id:string provider_charge_id:string = PaymentCharge;

postAddress#1e8caaeb street_line1:string street_line2:string city:string state:string country_iso2:string post_

paymentRequestedInfo#909c3f94 flags:# name:flags.0?string phone:flags.1?string email:flags.2?string shipping_ad

paymentSavedCredentialsCard#cdc27a1f id:string title:string = PaymentSavedCredentials;

webDocument#1c570ed1 url:string access_hash:long size:int mime_type:string attributes:Vector<DocumentAttribute>
webDocumentNoProxy#f9c8bcc6 url:string size:int mime_type:string attributes:Vector<DocumentAttribute> = WebDocu

inputWebDocument#9bed434d url:string size:int mime_type:string attributes:Vector<DocumentAttribute> = InputWebD

inputWebFileLocation#c239d686 url:string access_hash:long = InputWebFileLocation;
inputWebFileGeoPointLocation#9f2221c9 geo_point:InputGeoPoint access_hash:long w:int h:int zoom:int scale:int =

upload.webFile#21e753bc size:int mime_type:string file_type:storage.FileType mtime:int bytes:bytes = upload.Web

payments.paymentForm#3f56aea3 flags:# can_save_credentials:flags.2?true password_missing:flags.3?true bot_id:in

payments.validatedRequestedInfo#d1451883 flags:# id:flags.0?string shipping_options:flags.1?Vector<ShippingOpti

payments.paymentResult#4e5f810d updates:Updates = payments.PaymentResult;
payments.paymentVerificationNeeded#d8411139 url:string = payments.PaymentResult;

payments.paymentReceipt#500911e1 flags:# date:int bot_id:int invoice:Invoice provider_id:int info:flags.0?Payme

payments.savedInfo#fb8fe43c flags:# has_saved_credentials:flags.1?true saved_info:flags.0?PaymentRequestedInfo

inputPaymentCredentialsSaved#c10eb2cf id:string tmp_password:bytes = InputPaymentCredentials;
inputPaymentCredentials#3417d728 flags:# save:flags.0?true data:DataJSON = InputPaymentCredentials;
inputPaymentCredentialsApplePay#aa1c39f payment_data:DataJSON = InputPaymentCredentials;
```

```
inputPaymentCredentialsAndroidPay#ca05d50e payment_token:DataJSON google_transaction_id:string = InputPaymentCr

account.tmpPassword#db64fd34 tmp_password:bytes valid_until:int = account.TmpPassword;

shippingOption#b6213cdf id:string title:string prices:Vector<LabeledPrice> = ShippingOption;

inputStickerSetItem#ffa0a496 flags:# document:InputDocument emoji:string mask_coords:flags.0?MaskCoords = Input

inputPhoneCall#1e36fded id:long access_hash:long = InputPhoneCall;

phoneCallEmpty#5366c915 id:long = PhoneCall;
phoneCallWaiting#1b8f4ad1 flags:# video:flags.5?true id:long access_hash:long date:int admin_id:int participant
phoneCallRequested#87eabb53 flags:# video:flags.5?true id:long access_hash:long date:int admin_id:int participa
phoneCallAccepted#997c454a flags:# video:flags.5?true id:long access_hash:long date:int admin_id:int participan
phoneCall#8742ae7f flags:# p2p_allowed:flags.5?true id:long access_hash:long date:int admin_id:int participant_
phoneCallDiscarded#50ca4de1 flags:# need_rating:flags.2?true need_debug:flags.3?true video:flags.5?true id:long

phoneConnection#9d4c17c0 id:long ip:string ipv6:string port:int peer_tag:bytes = PhoneConnection;

phoneCallProtocol#a2bb35cb flags:# udp_p2p:flags.0?true udp_reflector:flags.1?true min_layer:int max_layer:int

phone.phoneCall#ec82e140 phone_call:PhoneCall users:Vector<User> = phone.PhoneCall;

upload.cdnFileReuploadNeeded#eea8e46e request_token:bytes = upload.CdnFile;
upload.cdnFile#a99fca4f bytes:bytes = upload.CdnFile;

cdnPublicKey#c982eaba dc_id:int public_key:string = CdnPublicKey;

cdnConfig#5725e40a public_keys:Vector<CdnPublicKey> = CdnConfig;

langPackString#cad181f6 key:string value:string = LangPackString;
langPackStringPluralized#6c47ac9f flags:# key:string zero_value:flags.0?string one_value:flags.1?string two_val
langPackStringDeleted#2979eeb2 key:string = LangPackString;

langPackDifference#f385c1f6 lang_code:string from_version:int version:int strings:Vector<LangPackString> = Lang

langPackLanguage#eeca5ce3 flags:# official:flags.0?true rtl:flags.2?true beta:flags.3?true name:string native_n

channelAdminLogEventActionChangeTitle#e6dfb825 prev_value:string new_value:string = ChannelAdminLogEventAction;
channelAdminLogEventActionChangeAbout#55188a2e prev_value:string new_value:string = ChannelAdminLogEventAction;
channelAdminLogEventActionChangeUsername#6a4afc38 prev_value:string new_value:string = ChannelAdminLogEventActi
channelAdminLogEventActionChangePhoto#434bd2af prev_photo:Photo new_photo:Photo = ChannelAdminLogEventAction;
channelAdminLogEventActionToggleInvites#1b7907ae new_value:Bool = ChannelAdminLogEventAction;
channelAdminLogEventActionToggleSignatures#26ae0971 new_value:Bool = ChannelAdminLogEventAction;
channelAdminLogEventActionUpdatePinned#e9e82c18 message:Message = ChannelAdminLogEventAction;
channelAdminLogEventActionEditMessage#709b2405 prev_message:Message new_message:Message = ChannelAdminLogEventA
channelAdminLogEventActionDeleteMessage#42e047bb message:Message = ChannelAdminLogEventAction;
channelAdminLogEventActionParticipantJoin#183040d3 = ChannelAdminLogEventAction;
channelAdminLogEventActionParticipantLeave#f89777f2 = ChannelAdminLogEventAction;
channelAdminLogEventActionParticipantInvite#e31c34d8 participant:ChannelParticipant = ChannelAdminLogEventActio
channelAdminLogEventActionParticipantToggleBan#e6d83d7e prev_participant:ChannelParticipant new_participant:Cha
channelAdminLogEventActionParticipantToggleAdmin#d5676710 prev_participant:ChannelParticipant new_participant:C
channelAdminLogEventActionChangeStickerSet#b1c3caa7 prev_stickerset:InputStickerSet new_stickerset:InputSticker
channelAdminLogEventActionTogglePreHistoryHidden#5f5c95f1 new_value:Bool = ChannelAdminLogEventAction;
channelAdminLogEventActionDefaultBannedRights#2df5fc0a prev_banned_rights:ChatBannedRights new_banned_rights:Ch
channelAdminLogEventActionStopPoll#8f079643 message:Message = ChannelAdminLogEventAction;
channelAdminLogEventActionChangeLinkedChat#a26f881b prev_value:int new_value:int = ChannelAdminLogEventAction;
channelAdminLogEventActionChangeLocation#e6b76ae prev_value:ChannelLocation new_value:ChannelLocation = Channel
channelAdminLogEventActionToggleSlowMode#53909779 prev_value:int new_value:int = ChannelAdminLogEventAction;

channelAdminLogEvent#3b5a3e40 id:long date:int user_id:int action:ChannelAdminLogEventAction = ChannelAdminLogE

channels.adminLogResults#ed8af74d events:Vector<ChannelAdminLogEvent> chats:Vector<Chat> users:Vector<User> = c

channelAdminLogEventsFilter#ea107ae4 flags:# join:flags.0?true leave:flags.1?true invite:flags.2?true ban:flags

popularContact#5ce14175 client_id:long importers:int = PopularContact;

messages.favedStickersNotModified#9e8fa6d3 = messages.FavedStickers;
messages.favedStickers#f37f2f16 hash:int packs:Vector<StickerPack> stickers:Vector<Document> = messages.FavedSt

recentMeUrlUnknown#46e1d13d url:string = RecentMeUrl;
recentMeUrlUser#8dbc3336 url:string user_id:int = RecentMeUrl;
recentMeUrlChat#a01b22f9 url:string chat_id:int = RecentMeUrl;
recentMeUrlChatInvite#eb49081d url:string chat_invite:ChatInvite = RecentMeUrl;
recentMeUrlStickerSet#bc0a57dc url:string set:StickerSetCovered = RecentMeUrl;

help.recentMeUrls#e0310d7 urls:Vector<RecentMeUrl> chats:Vector<Chat> users:Vector<User> = help.RecentMeUrls;

inputSingleMedia#1cc6e91f flags:# media:InputMedia random_id:long message:string entities:flags.0?Vector<Messag

webAuthorization#cac943f2 hash:long bot_id:int domain:string browser:string platform:string date_created:int da

account.webAuthorizations#ed56c9fc authorizations:Vector<WebAuthorization> users:Vector<User> = account.WebAuth

inputMessageID#a676a322 id:int = InputMessage;
inputMessageReplyTo#bad88395 id:int = InputMessage;
inputMessagePinned#86872538 = InputMessage;

inputDialogPeer#fcaafeb7 peer:InputPeer = InputDialogPeer;
inputDialogPeerFolder#64600527 folder_id:int = InputDialogPeer;

dialogPeer#e56dbf05 peer:Peer = DialogPeer;
dialogPeerFolder#514519e2 folder_id:int = DialogPeer;

messages.foundStickerSetsNotModified#d54b65d = messages.FoundStickerSets;
messages.foundStickerSets#5108d648 hash:int sets:Vector<StickerSetCovered> = messages.FoundStickerSets;

fileHash#6242c773 offset:int limit:int hash:bytes = FileHash;

inputClientProxy#75588b3f address:string port:int = InputClientProxy;

help.proxyDataEmpty#e09e1fb8 expires:int = help.ProxyData;
help.proxyDataPromo#2bf7ee23 expires:int peer:Peer chats:Vector<Chat> users:Vector<User> = help.ProxyData;

help.termsOfServiceUpdateEmpty#e3309f7f expires:int = help.TermsOfServiceUpdate;
help.termsOfServiceUpdate#28ecf961 expires:int terms_of_service:help.TermsOfService = help.TermsOfServiceUpdate

inputSecureFileUploaded#3334b0f0 id:long parts:int md5_checksum:string file_hash:bytes secret:bytes = InputSecu
inputSecureFile#5367e5be id:long access_hash:long = InputSecureFile;

secureFileEmpty#64199744 = SecureFile;
secureFile#e0277a62 id:long access_hash:long size:int dc_id:int date:int file_hash:bytes secret:bytes = SecureF

secureData#8aeabec3 data:bytes data_hash:bytes secret:bytes = SecureData;

securePlainPhone#7d6099dd phone:string = SecurePlainData;
securePlainEmail#21ec5a5f email:string = SecurePlainData;

secureValueTypePersonalDetails#9d2a81e3 = SecureValueType;
secureValueTypePassport#3dac6a00 = SecureValueType;
secureValueTypeDriverLicense#6e425c4 = SecureValueType;
secureValueTypeIdentityCard#a0d0744b = SecureValueType;
secureValueTypeInternalPassport#99a48f23 = SecureValueType;
secureValueTypeAddress#cbe31e26 = SecureValueType;
secureValueTypeUtilityBill#fc36954e = SecureValueType;
secureValueTypeBankStatement#89137c0d = SecureValueType;
secureValueTypeRentalAgreement#8b883488 = SecureValueType;
secureValueTypePassportRegistration#99e3806a = SecureValueType;
secureValueTypeTemporaryRegistration#ea02ec33 = SecureValueType;
secureValueTypePhone#b320aadb = SecureValueType;
```

```
secureValueTypeEmail#8e3ca7ee = SecureValueType;
secureValue#187fa0ca flags:# type:SecureValueType data:flags.0?SecureData front_side:flags.1?SecureFile reverse

inputSecureValue#db21d0a7 flags:# type:SecureValueType data:flags.0?SecureData front_side:flags.1?InputSecureFi

secureValueHash#ed1ecdb0 type:SecureValueType hash:bytes = SecureValueHash;

secureValueErrorData#e8a40bd9 type:SecureValueType data_hash:bytes field:string text:string = SecureValueError;
secureValueErrorFrontSide#be3dfa type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorReverseSide#868a2aa5 type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorSelfie#e537ced6 type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorFile#7a700873 type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorFiles#666220e9 type:SecureValueType file_hash:Vector<bytes> text:string = SecureValueError;
secureValueError#869d758f type:SecureValueType hash:bytes text:string = SecureValueError;
secureValueErrorTranslationFile#a1144770 type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorTranslationFiles#34636dd8 type:SecureValueType file_hash:Vector<bytes> text:string = SecureValu

secureCredentialsEncrypted#33f0ea47 data:bytes hash:bytes secret:bytes = SecureCredentialsEncrypted;

account.authorizationForm#ad2e1cd8 flags:# required_types:Vector<SecureRequiredType> values:Vector<SecureValue>

account.sentEmailCode#811f854f email_pattern:string length:int = account.SentEmailCode;

help.deepLinkInfoEmpty#66afa166 = help.DeepLinkInfo;
help.deepLinkInfo#6a4ee832 flags:# update_app:flags.0?true message:string entities:flags.1?Vector<MessageEntity

savedPhoneContact#1142bd56 phone:string first_name:string last_name:string date:int = SavedContact;

account.takeout#4dba4501 id:long = account.Takeout;

passwordKdfAlgoUnknown#d45ab096 = PasswordKdfAlgo;
passwordKdfAlgoSHA256SHA256PBKDF2HMACSHA512iter100000SHA256ModPow#3a912d4a salt1:bytes salt2:bytes g:int p:byte

securePasswordKdfAlgoUnknown#4a8537 = SecurePasswordKdfAlgo;
securePasswordKdfAlgoPBKDF2HMACSHA512iter100000#bbf2dda0 salt:bytes = SecurePasswordKdfAlgo;
securePasswordKdfAlgoSHA512#86471d92 salt:bytes = SecurePasswordKdfAlgo;

secureSecretSettings#1527bcac secure_algo:SecurePasswordKdfAlgo secure_secret:bytes secure_secret_id:long = Sec

inputCheckPasswordEmpty#9880f658 = InputCheckPasswordSRP;
inputCheckPasswordSRP#d27ff082 srp_id:long A:bytes M1:bytes = InputCheckPasswordSRP;

secureRequiredType#829d99da flags:# native_names:flags.0?true selfie_required:flags.1?true translation_required
secureRequiredTypeOneOf#27477b4 types:Vector<SecureRequiredType> = SecureRequiredType;

help.passportConfigNotModified#bfb9f457 = help.PassportConfig;
help.passportConfig#a098d6af hash:int countries_langs:DataJSON = help.PassportConfig;

inputAppEvent#1d1b1245 time:double type:string peer:long data:JSONValue = InputAppEvent;

jsonObjectValue#c0de1bd9 key:string value:JSONValue = JSONObjectValue;

jsonNull#3f6d7b68 = JSONValue;
jsonBool#c7345e6a value:Bool = JSONValue;
jsonNumber#2be0dfa4 value:double = JSONValue;
jsonString#b71e767a value:string = JSONValue;
jsonArray#f7444763 value:Vector<JSONValue> = JSONValue;
jsonObject#99c1d49d value:Vector<JSONObjectValue> = JSONValue;

pageTableCell#34566b6a flags:# header:flags.0?true align_center:flags.3?true align_right:flags.4?true valign_mi

pageTableRow#e0c0c5e5 cells:Vector<PageTableCell> = PageTableRow;

pageCaption#6f747657 text:RichText credit:RichText = PageCaption;

pageListItemText#b92fb6cd text:RichText = PageListItem;
pageListItemBlocks#25e073fc blocks:Vector<PageBlock> = PageListItem;

pageListOrderedItemText#5e068047 num:string text:RichText = PageListOrderedItem;
pageListOrderedItemBlocks#98dd8936 num:string blocks:Vector<PageBlock> = PageListOrderedItem;

pageRelatedArticle#b390dc08 flags:# url:string webpage_id:long title:flags.0?string description:flags.1?string

page#ae891bec flags:# part:flags.0?true rtl:flags.1?true v2:flags.2?true url:string blocks:Vector<PageBlock> ph

help.supportName#8c05f1c9 name:string = help.SupportName;

help.userInfoEmpty#f3ae2eed = help.UserInfo;
help.userInfo#1eb3758 message:string entities:Vector<MessageEntity> author:string date:int = help.UserInfo;

pollAnswer#6ca9c2e9 text:string option:bytes = PollAnswer;

poll#d5529d06 id:long flags:# closed:flags.0?true question:string answers:Vector<PollAnswer> = Poll;

pollAnswerVoters#3b6ddad2 flags:# chosen:flags.0?true option:bytes voters:int = PollAnswerVoters;

pollResults#5755785a flags:# min:flags.0?true results:flags.1?Vector<PollAnswerVoters> total_voters:flags.2?int

chatOnlines#f041e250 onlines:int = ChatOnlines;

statsURL#47a971e0 url:string = StatsURL;

chatAdminRights#5fb224d5 flags:# change_info:flags.0?true post_messages:flags.1?true edit_messages:flags.2?true

chatBannedRights#9f120418 flags:# view_messages:flags.0?true send_messages:flags.1?true send_media:flags.2?true

inputWallPaper#e630b979 id:long access_hash:long = InputWallPaper;
inputWallPaperSlug#72091c80 slug:string = InputWallPaper;

account.wallPapersNotModified#1c199183 = account.WallPapers;
account.wallPapers#702b65a9 hash:int wallpapers:Vector<WallPaper> = account.WallPapers;

codeSettings#debebe83 flags:# allow_flashcall:flags.0?true current_number:flags.1?true allow_app_hash:flags.4?t

wallPaperSettings#a12f40b8 flags:# blur:flags.1?true motion:flags.2?true background_color:flags.0?int intensity

autoDownloadSettings#d246fd47 flags:# disabled:flags.0?true video_preload_large:flags.1?true audio_preload_next

account.autoDownloadSettings#63cacf26 low:AutoDownloadSettings medium:AutoDownloadSettings high:AutoDownloadSet

emojiKeyword#d5b3b9f9 keyword:string emoticons:Vector<string> = EmojiKeyword;
emojiKeywordDeleted#236df622 keyword:string emoticons:Vector<string> = EmojiKeyword;

emojiKeywordsDifference#5cc761bd lang_code:string from_version:int version:int keywords:Vector<EmojiKeyword> =

emojiURL#a575739d url:string = EmojiURL;

emojiLanguage#b3fb5361 lang_code:string = EmojiLanguage;

fileLocationToBeDeprecated#bc7fc6cd volume_id:long local_id:int = FileLocation;

folder#ff544e65 flags:# autofill_new_broadcasts:flags.0?true autofill_public_groups:flags.1?true autofill_new_c

inputFolderPeer#fbd2c296 peer:InputPeer folder_id:int = InputFolderPeer;

folderPeer#e9baa668 peer:Peer folder_id:int = FolderPeer;

messages.searchCounter#e844ebff flags:# inexact:flags.1?true filter:MessagesFilter count:int = messages.SearchC

urlAuthResultRequest#92d33a0e flags:# request_write_access:flags.0?true bot:User domain:string = UrlAuthResult;
urlAuthResultAccepted#8f8c0e4e url:string = UrlAuthResult;
```

```
urlAuthResultDefault#a9d6db1f = UrlAuthResult;
channelLocationEmpty#bfb5ad8b = ChannelLocation;
channelLocation#209b82db geo_point:GeoPoint address:string = ChannelLocation;

peerLocated#ca461b5d peer:Peer expires:int distance:int = PeerLocated;

restrictionReason#d072acb4 platform:string reason:string text:string = RestrictionReason;

inputTheme#3c5693e9 id:long access_hash:long = InputTheme;
inputThemeSlug#f5890df1 slug:string = InputTheme;

themeDocumentNotModified#483d270c = Theme;
theme#f7d90ce0 flags:# creator:flags.0?true default:flags.1?true id:long access_hash:long slug:string title:str

account.themesNotModified#f41eb622 = account.Themes;
account.themes#7f676421 hash:int themes:Vector<Theme> = account.Themes;

---functions---

invokeAfterMsg#cb9f372d {X:Type} msg_id:long query:!X = X;
invokeAfterMsgs#3dc4b4f0 {X:Type} msg_ids:Vector<long> query:!X = X;
initConnection#785188b8 {X:Type} flags:# api_id:int device_model:string system_version:string app_version:strin
invokeWithLayer#da9b0d0d {X:Type} layer:int query:!X = X;
invokeWithoutUpdates#bf9459b7 {X:Type} query:!X = X;
invokeWithMessagesRange#365275f2 {X:Type} range:MessageRange query:!X = X;
invokeWithTakeout#aca9fd2e {X:Type} takeout_id:long query:!X = X;

auth.sendCode#a677244f phone_number:string api_id:int api_hash:string settings:CodeSettings = auth.SentCode;
auth.signUp#80eee427 phone_number:string phone_code_hash:string first_name:string last_name:string = auth.Autho
auth.signIn#bcd51581 phone_number:string phone_code_hash:string phone_code:string = auth.Authorization;
auth.logOut#5717da40 = Bool;
auth.resetAuthorizations#9fab0d1a = Bool;
auth.exportAuthorization#e5bfffcd dc_id:int = auth.ExportedAuthorization;
auth.importAuthorization#e3ef9613 id:int bytes:bytes = auth.Authorization;
auth.bindTempAuthKey#cdd42a05 perm_auth_key_id:long nonce:long expires_at:int encrypted_message:bytes = Bool;
auth.importBotAuthorization#67a3ff2c flags:int api_id:int api_hash:string bot_auth_token:string = auth.Authoriz
auth.checkPassword#d18b4d16 password:InputCheckPasswordSRP = auth.Authorization;
auth.requestPasswordRecovery#d897bc66 = auth.PasswordRecovery;
auth.recoverPassword#4ea56e92 code:string = auth.Authorization;
auth.resendCode#3ef1a9bf phone_number:string phone_code_hash:string = auth.SentCode;
auth.cancelCode#1f040578 phone_number:string phone_code_hash:string = Bool;
auth.dropTempAuthKeys#8e48a188 except_auth_keys:Vector<long> = Bool;

account.registerDevice#68976c6f flags:# no_muted:flags.0?true token_type:int token:string app_sandbox:Bool secr
account.unregisterDevice#3076c4bf token_type:int token:string other_uids:Vector<int> = Bool;
account.updateNotifySettings#84be5b93 peer:InputNotifyPeer settings:InputPeerNotifySettings = Bool;
account.getNotifySettings#12b3ad31 peer:InputNotifyPeer = PeerNotifySettings;
account.resetNotifySettings#db7e1747 = Bool;
account.updateProfile#78515775 flags:# first_name:flags.0?string last_name:flags.1?string about:flags.2?string
account.updateStatus#6628562c offline:Bool = Bool;
account.getWallPapers#aabb1763 hash:int = account.WallPapers;
account.reportPeer#ae189d5f peer:InputPeer reason:ReportReason = Bool;
account.checkUsername#2714d86c username:string = Bool;
account.updateUsername#3e0bdd7c username:string = User;
account.getPrivacy#dadbc950 key:InputPrivacyKey = account.PrivacyRules;
account.setPrivacy#c9f81ce8 key:InputPrivacyKey rules:Vector<InputPrivacyRule> = account.PrivacyRules;
account.deleteAccount#418d4e0b reason:string = Bool;
account.getAccountTTL#8fc711d = AccountDaysTTL;
account.setAccountTTL#2442485e ttl:AccountDaysTTL = Bool;
account.sendChangePhoneCode#82574ae5 phone_number:string settings:CodeSettings = auth.SentCode;
account.changePhone#70c32edb phone_number:string phone_code_hash:string phone_code:string = User;
account.updateDeviceLocked#38df3532 period:int = Bool;
account.getAuthorizations#e320c158 = account.Authorizations;
account.resetAuthorization#df77f3bc hash:long = Bool;
account.getPassword#548a30f5 = account.Password;
account.getPasswordSettings#9cd4eaf9 password:InputCheckPasswordSRP = account.PasswordSettings;
account.updatePasswordSettings#a59b102f password:InputCheckPasswordSRP new_settings:account.PasswordInputSettin
account.sendConfirmPhoneCode#1b3faa88 hash:string settings:CodeSettings = auth.SentCode;
account.confirmPhone#5f2178c3 phone_code_hash:string phone_code:string = Bool;
account.getTmpPassword#449e0b51 password:InputCheckPasswordSRP period:int = account.TmpPassword;
account.getWebAuthorizations#182e6d6f = account.WebAuthorizations;
account.resetWebAuthorization#2d01b9ef hash:long = Bool;
account.resetWebAuthorizations#682d2594 = Bool;
account.getAllSecureValues#b288bc7d = Vector<SecureValue>;
account.getSecureValue#73665bc2 types:Vector<SecureValueType> = Vector<SecureValue>;
account.saveSecureValue#899fe31d value:InputSecureValue secure_secret_id:long = SecureValue;
account.deleteSecureValue#b880bc4b types:Vector<SecureValueType> = Bool;
account.getAuthorizationForm#b86ba8e1 bot_id:int scope:string public_key:string = account.AuthorizationForm;
account.acceptAuthorization#e7027c94 bot_id:int scope:string public_key:string value_hashes:Vector<SecureValueH
account.sendVerifyPhoneCode#a5a356f9 phone_number:string settings:CodeSettings = auth.SentCode;
account.verifyPhone#4dd3a7f6 phone_number:string phone_code_hash:string phone_code:string = Bool;
account.sendVerifyEmailCode#7011509f email:string = account.SentEmailCode;
account.verifyEmail#ecba39db email:string code:string = Bool;
account.initTakeoutSession#f05b4804 flags:# contacts:flags.0?true message_users:flags.1?true message_chats:flag
account.finishTakeoutSession#1d2652ee flags:# success:flags.0?true = Bool;
account.confirmPasswordEmail#8fdf1920 code:string = Bool;
account.resendPasswordEmail#7a7f2a15 = Bool;
account.cancelPasswordEmail#c1cbd5b6 = Bool;
account.getContactSignUpNotification#9f07c728 = Bool;
account.setContactSignUpNotification#cff43f61 silent:Bool = Bool;
account.getNotifyExceptions#53577479 flags:# compare_sound:flags.1?true peer:flags.0?InputNotifyPeer = Updates;
account.getWallPaper#fc8ddbea wallpaper:InputWallPaper = WallPaper;
account.uploadWallPaper#dd853661 file:InputFile mime_type:string settings:WallPaperSettings = WallPaper;
account.saveWallPaper#6c5a5b37 wallpaper:InputWallPaper unsave:Bool settings:WallPaperSettings = Bool;
account.installWallPaper#feed5769 wallpaper:InputWallPaper settings:WallPaperSettings = Bool;
account.resetWallPapers#bb3b9804 = Bool;
account.getAutoDownloadSettings#56da0b3f = account.AutoDownloadSettings;
account.saveAutoDownloadSettings#76f36233 flags:# low:flags.0?true high:flags.1?true settings:AutoDownloadSetti
account.uploadTheme#1c3db333 flags:# file:InputFile thumb:flags.0?InputFile file_name:string mime_type:string =
account.createTheme#2b7ffd7f slug:string title:string document:InputDocument = Theme;
account.updateTheme#3b8ea202 flags:# format:string theme:InputTheme slug:flags.0?string title:flags.1?string do
account.saveTheme#f257106c theme:InputTheme unsave:Bool = Bool;
account.installTheme#7ae43737 flags:# dark:flags.0?true format:flags.1?string theme:flags.1?InputTheme = Bool;
account.getTheme#8d9d742b format:string theme:InputTheme document_id:long = Theme;
account.getThemes#285946f8 format:string hash:int = account.Themes;

users.getUsers#d91a548 id:Vector<InputUser> = Vector<User>;
users.getFullUser#ca30a5b1 id:InputUser = UserFull;
users.setSecureValueErrors#90c894b5 id:InputUser errors:Vector<SecureValueError> = Bool;

contacts.getContactIDs#2caa4a42 hash:int = Vector<int>;
contacts.getStatuses#c4a353ee = Vector<ContactStatus>;
contacts.getContacts#c023849f hash:int = contacts.Contacts;
contacts.importContacts#2c800be5 contacts:Vector<InputContact> = contacts.ImportedContacts;
contacts.deleteContacts#96a0e00 id:Vector<InputUser> = Updates;
contacts.deleteByPhones#1013fd9e phones:Vector<string> = Bool;
contacts.block#332b49fc id:InputUser = Bool;
contacts.unblock#e54100bd id:InputUser = Bool;
contacts.getBlocked#f57c350f offset:int limit:int = contacts.Blocked;
contacts.search#11f812d8 q:string limit:int = contacts.Found;
contacts.resolveUsername#f93ccba3 username:string = contacts.ResolvedPeer;
contacts.getTopPeers#d4982db5 flags:# correspondents:flags.0?true bots_pm:flags.1?true bots_inline:flags.2?true
contacts.resetTopPeerRating#1ae373ac category:TopPeerCategory peer:InputPeer = Bool;
contacts.resetSaved#879537f1 = Bool;
contacts.getSaved#82f1e39f = Vector<SavedContact>;
contacts.toggleTopPeers#8514bdda enabled:Bool = Bool;
contacts.addContact#e8f463d0 flags:# add_phone_privacy_exception:flags.0?true id:InputUser first_name:string la
contacts.acceptContact#f831a20f id:InputUser = Updates;
contacts.getLocated#a356056 geo_point:InputGeoPoint = Updates;
```

```
messages.getMessages#63c66506 id:Vector<InputMessage> = messages.Messages;
messages.getDialogs#a0ee3b73 flags:# exclude_pinned:flags.0?true folder_id:flags.1?int offset_date:int offset_i
messages.getHistory#dcbb8260 peer:InputPeer offset_id:int offset_date:int add_offset:int limit:int max_id:int m
messages.search#8614ef68 flags:# peer:InputPeer q:string from_id:flags.0?InputUser filter:MessagesFilter min_da
messages.readHistory#e306d3a peer:InputPeer max_id:int = messages.AffectedMessages;
messages.deleteHistory#1c015b09 flags:# just_clear:flags.0?true revoke:flags.1?true peer:InputPeer max_id:int =
messages.deleteMessages#e58e95d2 flags:# revoke:flags.0?true id:Vector<int> = messages.AffectedMessages;
messages.receivedMessages#5a954c0 max_id:int = Vector<ReceivedNotifyMessage>;
messages.setTyping#a3825e50 peer:InputPeer action:SendMessageAction = Bool;
messages.sendMessage#520c3870 flags:# no_webpage:flags.1?true silent:flags.5?true background:flags.6?true clear
messages.sendMedia#3491eba9 flags:# silent:flags.5?true background:flags.6?true clear_draft:flags.7?true peer:I
messages.forwardMessages#d9fee60e flags:# silent:flags.5?true background:flags.6?true with_my_score:flags.8?tru
messages.reportSpam#cf1592db peer:InputPeer = Bool;
messages.getPeerSettings#3672e09c peer:InputPeer = PeerSettings;
messages.report#bd82b658 peer:InputPeer id:Vector<int> reason:ReportReason = Bool;
messages.getChats#3c6aa187 id:Vector<int> = messages.Chats;
messages.getFullChat#3b831c66 chat_id:int = messages.ChatFull;
messages.editChatTitle#dc452855 chat_id:int title:string = Updates;
messages.editChatPhoto#ca4c79d8 chat_id:int photo:InputChatPhoto = Updates;
messages.addChatUser#f9a0aa09 chat_id:int user_id:InputUser fwd_limit:int = Updates;
messages.deleteChatUser#e0611f16 chat_id:int user_id:InputUser = Updates;
messages.createChat#9cb126e users:Vector<InputUser> title:string = Updates;
messages.getDhConfig#26cf8950 version:int random_length:int = messages.DhConfig;
messages.requestEncryption#f64daf43 user_id:InputUser random_id:int g_a:bytes = EncryptedChat;
messages.acceptEncryption#3dbc0415 peer:InputEncryptedChat g_b:bytes key_fingerprint:long = EncryptedChat;
messages.discardEncryption#edd923c5 chat_id:int = Bool;
messages.setEncryptedTyping#791451ed peer:InputEncryptedChat typing:Bool = Bool;
messages.readEncryptedHistory#7f4b690a peer:InputEncryptedChat max_date:int = Bool;
messages.sendEncrypted#a9776773 peer:InputEncryptedChat random_id:long data:bytes = messages.SentEncryptedMessa
messages.sendEncryptedFile#9a901b66 peer:InputEncryptedChat random_id:long data:bytes file:InputEncryptedFile =
messages.sendEncryptedService#32d439a4 peer:InputEncryptedChat random_id:long data:bytes = messages.SentEncrypt
messages.receivedQueue#55a5bb66 max_qts:int = Vector<long>;
messages.reportEncryptedSpam#4b0c8c0f peer:InputEncryptedChat = Bool;
messages.readMessageContents#36a73f77 id:Vector<int> = messages.AffectedMessages;
messages.getStickers#43d4f2c emoticon:string hash:int = messages.Stickers;
messages.getAllStickers#1c9618b1 hash:int = messages.AllStickers;
messages.getWebPagePreview#8b68b0cc flags:# message:string entities:flags.3?Vector<MessageEntity> = MessageMedi
messages.exportChatInvite#df7534c peer:InputPeer = ExportedChatInvite;
messages.checkChatInvite#3eadb1bb hash:string = ChatInvite;
messages.importChatInvite#6c50051c hash:string = Updates;
messages.getStickerSet#2619a90e stickerset:InputStickerSet = messages.StickerSet;
messages.installStickerSet#c78fe460 stickerset:InputStickerSet archived:Bool = messages.StickerSetInstallResult
messages.uninstallStickerSet#f96e55de stickerset:InputStickerSet = Bool;
messages.startBot#e6df7378 bot:InputUser peer:InputPeer random_id:long start_param:string = Updates;
messages.getMessagesViews#c4c8a55d peer:InputPeer id:Vector<int> increment:Bool = Vector<int>;
messages.editChatAdmin#a9e69f2e chat_id:int user_id:InputUser is_admin:Bool = Bool;
messages.migrateChat#15a3b8e3 chat_id:int = Updates;
messages.searchGlobal#bf7225a4 flags:# folder_id:flags.0?int q:string offset_rate:int offset_peer:InputPeer off
messages.reorderStickerSets#78337739 flags:# masks:flags.0?true order:Vector<long> = Bool;
messages.getDocumentByHash#338e2464 sha256:bytes size:int mime_type:string = Document;
messages.searchGifs#bf9a776b q:string offset:int = messages.FoundGifs;
messages.getSavedGifs#83bf3d52 hash:int = messages.SavedGifs;
messages.saveGif#327a30cb id:InputDocument unsave:Bool = Bool;
messages.getInlineBotResults#514e999d flags:# bot:InputUser peer:InputPeer geo_point:flags.0?InputGeoPoint quer
messages.setInlineBotResults#eb5ea206 flags:# gallery:flags.0?true private:flags.1?true query_id:long results:V
messages.sendInlineBotResult#220815b0 flags:# silent:flags.5?true background:flags.6?true clear_draft:flags.7?t
messages.getMessageEditData#fda68d36 peer:InputPeer id:int = messages.MessageEditData;
messages.editMessage#48f71778 flags:# no_webpage:flags.1?true peer:InputPeer id:int message:flags.11?string med
messages.editInlineBotMessage#83557dba flags:# no_webpage:flags.1?true id:InputBotInlineMessageID message:flags
messages.getBotCallbackAnswer#810a9fec flags:# game:flags.1?true peer:InputPeer msg_id:int data:flags.0?bytes =
messages.setBotCallbackAnswer#d58f130a flags:# alert:flags.1?true query_id:long message:flags.0?string url:flag
messages.getPeerDialogs#e470bcfd peers:Vector<InputDialogPeer> = messages.PeerDialogs;
messages.saveDraft#bc39e14b flags:# no_webpage:flags.1?true reply_to_msg_id:flags.0?int peer:InputPeer message:
messages.getAllDrafts#6a3f8d65 = Updates;
messages.getFeaturedStickers#2dacca4f hash:int = messages.FeaturedStickers;
messages.readFeaturedStickers#5b118126 id:Vector<long> = Bool;
messages.getRecentStickers#5ea192c9 flags:# attached:flags.0?true hash:int = messages.RecentStickers;
messages.saveRecentSticker#392718f8 flags:# attached:flags.0?true id:InputDocument unsave:Bool = Bool;
messages.clearRecentStickers#8999602d flags:# attached:flags.0?true = Bool;
messages.getArchivedStickers#57f17692 flags:# masks:flags.0?true offset_id:long limit:int = messages.ArchivedSt
messages.getMaskStickers#65b8c79f hash:int = messages.AllStickers;
messages.getAttachedStickers#cc5b67cc media:InputStickeredMedia = Vector<StickerSetCovered>;
messages.setGameScore#8ef8ecc0 flags:# edit_message:flags.0?true force:flags.1?true peer:InputPeer id:int user_
messages.setInlineGameScore#15ad9f64 flags:# edit_message:flags.0?true force:flags.1?true id:InputBotInlineMess
messages.getGameHighScores#e822649d peer:InputPeer id:int user_id:InputUser = messages.HighScores;
messages.getInlineGameHighScores#f635e1b id:InputBotInlineMessageID user_id:InputUser = messages.HighScores;
messages.getCommonChats#d0a48c4 user_id:InputUser max_id:int limit:int = messages.Chats;
messages.getAllChats#eba80ff0 except_ids:Vector<int> = messages.Chats;
messages.getWebPage#32ca8f91 url:string hash:int = WebPage;
messages.toggleDialogPin#a731e257 flags:# pinned:flags.0?true peer:InputDialogPeer = Bool;
messages.reorderPinnedDialogs#3b1adf37 flags:# force:flags.0?true folder_id:int order:Vector<InputDialogPeer> =
messages.getPinnedDialogs#d6b94df2 folder_id:int = messages.PeerDialogs;
messages.setBotShippingResults#e5f672fa flags:# query_id:long error:flags.0?string shipping_options:flags.1?Vec
messages.setBotPrecheckoutResults#9c2dd95 flags:# success:flags.1?true query_id:long error:flags.0?string = Boo
messages.uploadMedia#519bc2b1 peer:InputPeer media:InputMedia = MessageMedia;
messages.sendScreenshotNotification#c97df020 peer:InputPeer reply_to_msg_id:int random_id:long = Updates;
messages.getFavedStickers#21ce0b0e hash:int = messages.FavedStickers;
messages.faveSticker#b9ffc55b id:InputDocument unfave:Bool = Bool;
messages.getUnreadMentions#46578472 peer:InputPeer offset_id:int add_offset:int limit:int max_id:int min_id:int
messages.readMentions#f0189d3 peer:InputPeer = messages.AffectedHistory;
messages.getRecentLocations#bbc45b09 peer:InputPeer limit:int hash:int = messages.Messages;
messages.sendMultiMedia#cc0110cb flags:# silent:flags.5?true background:flags.6?true clear_draft:flags.7?true p
messages.uploadEncryptedFile#5057c497 peer:InputEncryptedChat file:InputEncryptedFile = EncryptedFile;
messages.searchStickerSets#c2b7d08b flags:# exclude_featured:flags.0?true q:string hash:int = messages.FoundSti
messages.getSplitRanges#1cff7e08 = Vector<MessageRange>;
messages.markDialogUnread#c286d98f flags:# unread:flags.0?true peer:InputDialogPeer = Bool;
messages.getDialogUnreadMarks#22e24e22 = Vector<DialogPeer>;
messages.clearAllDrafts#7e58ee9c = Bool;
messages.updatePinnedMessage#d2aaf7ec flags:# silent:flags.0?true peer:InputPeer id:int = Updates;
messages.sendVote#10ea6184 peer:InputPeer msg_id:int options:Vector<bytes> = Updates;
messages.getPollResults#73bb643b peer:InputPeer msg_id:int = Updates;
messages.getOnlines#6e2be050 peer:InputPeer = ChatOnlines;
messages.getStatsURL#812c2ae6 flags:# dark:flags.0?true peer:InputPeer params:string = StatsURL;
messages.editChatAbout#def60797 peer:InputPeer about:string = Bool;
messages.editChatDefaultBannedRights#a5866b41 peer:InputPeer banned_rights:ChatBannedRights = Updates;
messages.getEmojiKeywords#35a0e062 lang_code:string = EmojiKeywordsDifference;
messages.getEmojiKeywordsDifference#1508b6af lang_code:string from_version:int = EmojiKeywordsDifference;
messages.getEmojiKeywordsLanguages#4e9963b2 lang_codes:Vector<string> = Vector<EmojiLanguage>;
messages.getEmojiURL#d5b10c26 lang_code:string = EmojiURL;
messages.getSearchCounters#732eef00 peer:InputPeer filters:Vector<MessagesFilter> = Vector<messages.SearchCount
messages.requestUrlAuth#e33f5613 peer:InputPeer msg_id:int button_id:int = UrlAuthResult;
messages.acceptUrlAuth#f729ea98 flags:# write_allowed:flags.0?true peer:InputPeer msg_id:int button_id:int = Ur
messages.hidePeerSettingsBar#4facb138 peer:InputPeer = Bool;
messages.getScheduledHistory#e2c2685b peer:InputPeer hash:int = messages.Messages;
messages.getScheduledMessages#bdbb0464 peer:InputPeer id:Vector<int> = messages.Messages;
messages.sendScheduledMessages#bd38850a peer:InputPeer id:Vector<int> = Updates;
messages.deleteScheduledMessages#59ae2b16 peer:InputPeer id:Vector<int> = Updates;

updates.getState#edd4882a = updates.State;
updates.getDifference#25939651 flags:# pts:int pts_total_limit:flags.0?int date:int qts:int = updates.Differenc
updates.getChannelDifference#3173d78 flags:# force:flags.0?true channel:InputChannel filter:ChannelMessagesFilt

photos.updateProfilePhoto#f0bb5152 id:InputPhoto = UserProfilePhoto;
photos.uploadProfilePhoto#4f32c098 file:InputFile = photos.Photo;
photos.deletePhotos#87cf7f2f id:Vector<InputPhoto> = Vector<long>;
photos.getUserPhotos#91cd32a8 user_id:InputUser offset:int max_id:long limit:int = photos.Photos;

upload.saveFilePart#b304a621 file_id:long file_part:int bytes:bytes = Bool;
upload.getFile#b15a9afc flags:# precise:flags.0?true location:InputFileLocation offset:int limit:int = upload.F
upload.saveBigFilePart#de7b673d file_id:long file_part:int file_total_parts:int bytes:bytes = Bool;
upload.getWebFile#24e6818d location:InputWebFileLocation offset:int limit:int = upload.WebFile;
upload.getCdnFile#2000bcc3 file_token:bytes offset:int limit:int = upload.CdnFile;
```

```
upload.reuploadCdnFile#9b2754a8 file_token:bytes request_token:bytes = Vector<FileHash>;
upload.getCdnFileHashes#4da54231 file_token:bytes offset:int = Vector<FileHash>;
upload.getFileHashes#c7025931 location:InputFileLocation offset:int = Vector<FileHash>;

help.getConfig#c4f9186b = Config;
help.getNearestDc#1fb33026 = NearestDc;
help.getAppUpdate#522d5a7d source:string = help.AppUpdate;
help.getInviteText#4d392343 = help.InviteText;
help.getSupport#9cdf08cd = help.Support;
help.getAppChangelog#9010ef6f prev_app_version:string = Updates;
help.setBotUpdatesStatus#ec22cfcd pending_updates_count:int message:string = Bool;
help.getCdnConfig#52029342 = CdnConfig;
help.getRecentMeUrls#3dc0f114 referer:string = help.RecentMeUrls;
help.getProxyData#3d7758e1 = help.ProxyData;
help.getTermsOfServiceUpdate#2ca51fd1 = help.TermsOfServiceUpdate;
help.acceptTermsOfService#ee72f79a id:DataJSON = Bool;
help.getDeepLinkInfo#3fedc75f path:string = help.DeepLinkInfo;
help.getAppConfig#98914110 = JSONValue;
help.saveAppLog#6f02f748 events:Vector<InputAppEvent> = Bool;
help.getPassportConfig#c661ad08 hash:int = help.PassportConfig;
help.getSupportName#d360e72c = help.SupportName;
help.getUserInfo#38a08d3 user_id:InputUser = help.UserInfo;
help.editUserInfo#66b91b70 user_id:InputUser message:string entities:Vector<MessageEntity> = help.UserInfo;

channels.readHistory#cc104937 channel:InputChannel max_id:int = Bool;
channels.deleteMessages#84c1fd4e channel:InputChannel id:Vector<int> = messages.AffectedMessages;
channels.deleteUserHistory#d10dd71b channel:InputChannel user_id:InputUser = messages.AffectedHistory;
channels.reportSpam#fe087810 channel:InputChannel user_id:InputUser id:Vector<int> = Bool;
channels.getMessages#ad8c9a23 channel:InputChannel id:Vector<InputMessage> = messages.Messages;
channels.getParticipants#123e05e9 channel:InputChannel filter:ChannelParticipantsFilter offset:int limit:int ha
channels.getParticipant#546dd7a6 channel:InputChannel user_id:InputUser = channels.ChannelParticipant;
channels.getChannels#a7f6bbb id:Vector<InputChannel> = messages.Chats;
channels.getFullChannel#8736a09 channel:InputChannel = messages.ChatFull;
channels.createChannel#3d5fb10f flags:# broadcast:flags.0?true megagroup:flags.1?true title:string about:string
channels.editAdmin#d33c8902 channel:InputChannel user_id:InputUser admin_rights:ChatAdminRights rank:string = U
channels.editTitle#566decd0 channel:InputChannel title:string = Updates;
channels.editPhoto#f12e57c9 channel:InputChannel photo:InputChatPhoto = Updates;
channels.checkUsername#10e6bd2c channel:InputChannel username:string = Bool;
channels.updateUsername#3514b3de channel:InputChannel username:string = Bool;
channels.joinChannel#24b524c5 channel:InputChannel = Updates;
channels.leaveChannel#f836aa95 channel:InputChannel = Updates;
channels.inviteToChannel#199f3a6c channel:InputChannel users:Vector<InputUser> = Updates;
channels.deleteChannel#c0111fe3 channel:InputChannel = Updates;
channels.exportMessageLink#ceb77163 channel:InputChannel id:int grouped:Bool = ExportedMessageLink;
channels.toggleSignatures#1f69b606 channel:InputChannel enabled:Bool = Updates;
channels.getAdminedPublicChannels#f8b036af flags:# by_location:flags.0?true check_limit:flags.1?true = messages
channels.editBanned#72796912 channel:InputChannel user_id:InputUser banned_rights:ChatBannedRights = Updates;
channels.getAdminLog#33ddf480 flags:# channel:InputChannel q:string events_filter:flags.0?ChannelAdminLogEvents
channels.setStickers#ea8ca4f9 channel:InputChannel stickerset:InputStickerSet = Bool;
channels.readMessageContents#eab5dc38 channel:InputChannel id:Vector<int> = Bool;
channels.deleteHistory#af369d42 channel:InputChannel max_id:int = Bool;
channels.togglePreHistoryHidden#eabbb94c channel:InputChannel enabled:Bool = Updates;
channels.getLeftChannels#8341ecc0 offset:int = messages.Chats;
channels.getGroupsForDiscussion#f5dad378 = messages.Chats;
channels.setDiscussionGroup#40582bb2 broadcast:InputChannel group:InputChannel = Bool;
channels.editCreator#8f38cd1f channel:InputChannel user_id:InputUser password:InputCheckPasswordSRP = Updates;
channels.editLocation#58e63f6d channel:InputChannel geo_point:InputGeoPoint address:string = Bool;
channels.toggleSlowMode#edd49ef0 channel:InputChannel seconds:int = Updates;

bots.sendCustomRequest#aa2769ed custom_method:string params:DataJSON = DataJSON;
bots.answerWebhookJSONQuery#e6213f4d query_id:long data:DataJSON = Bool;

payments.getPaymentForm#99f09745 msg_id:int = payments.PaymentForm;
payments.getPaymentReceipt#a092a980 msg_id:int = payments.PaymentReceipt;
payments.validateRequestedInfo#770a8e74 flags:# save:flags.0?true msg_id:int info:PaymentRequestedInfo = paymen
payments.sendPaymentForm#2b8879b3 flags:# msg_id:int requested_info_id:flags.0?string shipping_option_id:flags.
payments.getSavedInfo#227d824b = payments.SavedInfo;
payments.clearSavedInfo#d83d70c1 flags:# credentials:flags.0?true info:flags.1?true = Bool;

stickers.createStickerSet#9bd86e6a flags:# masks:flags.0?true user_id:InputUser title:string short_name:string
stickers.removeStickerFromSet#f7760f51 sticker:InputDocument = messages.StickerSet;
stickers.changeStickerPosition#ffb6d4ca sticker:InputDocument position:int = messages.StickerSet;
stickers.addStickerToSet#8653febe stickerset:InputStickerSet sticker:InputStickerSetItem = messages.StickerSet;

phone.getCallConfig#55451fa9 = DataJSON;
phone.requestCall#42ff96ed flags:# video:flags.0?true user_id:InputUser random_id:int g_a_hash:bytes protocol:P
phone.acceptCall#3bd2b4a0 peer:InputPhoneCall g_b:bytes protocol:PhoneCallProtocol = phone.PhoneCall;
phone.confirmCall#2efe1722 peer:InputPhoneCall g_a:bytes key_fingerprint:long protocol:PhoneCallProtocol = phon
phone.receivedCall#17d54f61 peer:InputPhoneCall = Bool;
phone.discardCall#b2cbc1c0 flags:# video:flags.0?true peer:InputPhoneCall duration:int reason:PhoneCallDiscardR
phone.setCallRating#59ead627 flags:# user_initiative:flags.0?true peer:InputPhoneCall rating:int comment:string
phone.saveCallDebug#277add7e peer:InputPhoneCall debug:DataJSON = Bool;

langpack.getLangPack#f2f2330a lang_pack:string lang_code:string = LangPackDifference;
langpack.getStrings#efea3803 lang_pack:string lang_code:string keys:Vector<string> = Vector<LangPackString>;
langpack.getDifference#cd984aa5 lang_pack:string lang_code:string from_version:int = LangPackDifference;
langpack.getLanguages#42c6978f lang_pack:string = Vector<LangPackLanguage>;
langpack.getLanguage#6a596502 lang_pack:string lang_code:string = LangPackLanguage;

folders.editPeerFolders#6847d0ab folder_peers:Vector<InputFolderPeer> = Updates;
folders.deleteFolder#1c295881 folder_id:int = Updates;
```

Layers

Below you will find information on scheme changes. For more details on the use of layers, see Invoking API methods.

Layer 105

The API underwent huge changes, a full reread of the documentation is required.

Layer 23

Scheme changes

New Methods

- Added account.getPrivacy – Get privacy settings of current account
- Added account.setPrivacy – Change privacy settings of current account
- Added account.deleteAccount – Delete the user's account from the telegram servers. Can be used, for example, to delete the account of a user that provided the login code, but forgot the 2FA password and no recovery method is configured.
- Added account.getAccountTTL – Get days to live of account
- Added account.setAccountTTL – Set account self-destruction period
- Added invokeWithLayer – Invoke the specified query using the specified API layer
- Added contacts.resolveUsername – Resolve a @username to get peer info
- Added account.sendChangePhoneCode – Verify a new phone number to associate to the current account
- Added account.changePhone – Change the phone number of the current account
- Added messages.getStickers – Get stickers by emoji
- Added messages.getAllStickers – Get all installed stickers
- Added account.updateDeviceLocked – When client-side passcode lock feature is enabled, will not show message texts in incoming PUSH notifications.

Deleted Methods

- Removed invokeWithLayer18

New Constructors

- Added userStatusRecently – Online status: last seen recently
- Added userStatusLastWeek – Online status: last seen last week
- Added userStatusLastMonth – Online status: last seen last month
- Added updatePrivacy – Privacy rules were changed
- Added inputPrivacyKeyStatusTimestamp – Whether we can see the exact last online timestamp of the user
- Added privacyKeyStatusTimestamp – Whether we can see the last online timestamp
- Added inputPrivacyValueAllowContacts – Allow only contacts
- Added inputPrivacyValueAllowAll – Allow all users
- Added inputPrivacyValueAllowUsers – Allow only certain users
- Added inputPrivacyValueDisallowContacts – Disallow only contacts
- Added inputPrivacyValueDisallowAll – Disallow all
- Added inputPrivacyValueDisallowUsers – Disallow only certain users
- Added privacyValueAllowContacts – Allow all contacts
- Added privacyValueAllowAll – Allow all users
- Added privacyValueAllowUsers – Allow only certain users
- Added privacyValueDisallowContacts – Disallow only contacts
- Added privacyValueDisallowAll – Disallow all users
- Added privacyValueDisallowUsers – Disallow only certain users
- Added account.privacyRules – Privacy rules
- Added accountDaysTTL – Time to live in days of the current account
- Added account.sentChangePhoneCode
- Added updateUserPhone – A user's phone number was changed
- Added documentAttributeImageSize – Defines the width and height of an image uploaded as document
- Added documentAttributeAnimated – Defines an animated GIF
- Added documentAttributeSticker – Defines a sticker
- Added documentAttributeVideo – Defines a video
- Added documentAttributeAudio – Represents an audio file
- Added documentAttributeFilename – A simple document with a file name
- Added messages.stickersNotModified – No new stickers were found for the given query
- Added messages.stickers – Found stickers
- Added stickerPack – A stickerpack is a group of stickers associated to the same emoji.
  It is **not** a sticker pack the way it is usually intended, you may be looking for a StickerSet.
- Added messages.allStickersNotModified – Info about all installed stickers hasn't changed
- Added messages.allStickers – Info about all installed stickers
- Added disabledFeature

Changed Constructors

- Added **status** parameter, removed **expires** parameter in contactStatus
- Added **expires**, **chat_big_size**, **disabled_features** parameters in config
- Added **attributes** parameter, removed **file_name** parameter in inputMediaUploadedDocument
- Added **attributes** parameter, removed **file_name** parameter in inputMediaUploadedThumbDocument
- Added **attributes** parameter, removed **user_id**, **file_name** parameters in document

Scheme

```
contactStatus#d3680c61 user_id:int status:UserStatus = ContactStatus;
config#7dae33e0 date:int expires:int test_mode:Bool this_dc:int dc_options:Vector<DcOption> chat_big_size:int c

inputMediaUploadedDocument#ffe76b78 file:InputFile mime_type:string attributes:Vector<DocumentAttribute> = Inpu
inputMediaUploadedThumbDocument#41481486 file:InputFile thumb:InputFile mime_type:string attributes:Vector<Docu

document#f9a39f4f id:long access_hash:long date:int mime_type:string size:int thumb:PhotoSize dc_id:int attribu

userStatusRecently#e26f42f1 = UserStatus;
userStatusLastWeek#07bf09fc = UserStatus;
userStatusLastMonth#77ebc742 = UserStatus;
updatePrivacy#ee3b272a key:PrivacyKey rules:Vector<PrivacyRule> = Update;

inputPrivacyKeyStatusTimestamp#4f96cb18 = InputPrivacyKey;

privacyKeyStatusTimestamp#bc2eab30 = PrivacyKey;

inputPrivacyValueAllowContacts#0d09e07b = InputPrivacyRule;
inputPrivacyValueAllowAll#184b35ce = InputPrivacyRule;
inputPrivacyValueAllowUsers#131cc67f users:Vector<InputUser> = InputPrivacyRule;
inputPrivacyValueDisallowContacts#0ba52007 = InputPrivacyRule;
inputPrivacyValueDisallowAll#d66b66c9 = InputPrivacyRule;
inputPrivacyValueDisallowUsers#90110467 users:Vector<InputUser> = InputPrivacyRule;

privacyValueAllowContacts#fffe1bac = PrivacyRule;
privacyValueAllowAll#65427b82 = PrivacyRule;
privacyValueAllowUsers#4d5bbe0c users:Vector<int> = PrivacyRule;
```

```
privacyValueDisallowContacts#f888fa1a = PrivacyRule;
privacyValueDisallowAll#8b73e763 = PrivacyRule;
privacyValueDisallowUsers#0c7f49b7 users:Vector<int> = PrivacyRule;

account.privacyRules#554abb6f rules:Vector<PrivacyRule> users:Vector<User> = account.PrivacyRules;
accountDaysTTL#b8d0afdf days:int = AccountDaysTTL;
account.sentChangePhoneCode#a4f58c4c phone_code_hash:string send_call_timeout:int = account.SentChangePhoneCode

updateUserPhone#12b9417b user_id:int phone:string = Update;

documentAttributeImageSize#6c37c15c w:int h:int = DocumentAttribute;
documentAttributeAnimated#11b58939 = DocumentAttribute;
documentAttributeSticker#fb0a5727 = DocumentAttribute;
documentAttributeVideo#5910cccb duration:int w:int h:int = DocumentAttribute;
documentAttributeAudio#051448e5 duration:int = DocumentAttribute;
documentAttributeFilename#15590068 file_name:string = DocumentAttribute;

messages.stickersNotModified#f1749a22 = messages.Stickers;
messages.stickers#8a8ecd32 hash:string stickers:Vector<Document> = messages.Stickers;

stickerPack#12b299d4 emoticon:string documents:Vector<long> = StickerPack;

messages.allStickersNotModified#e86602c3 = messages.AllStickers;
messages.allStickers#dcef3102 hash:string packs:Vector<StickerPack> documents:Vector<Document> = messages.AllSt

disabledFeature#ae636f24 feature:string description:string = DisabledFeature;

---functions---

account.getPrivacy#dadbc950 key:InputPrivacyKey = account.PrivacyRules;
account.setPrivacy#c9f81ce8 key:InputPrivacyKey rules:Vector<InputPrivacyRule> = account.PrivacyRules;
account.deleteAccount#418d4e0b reason:string = Bool;
account.getAccountTTL#08fc711d = AccountDaysTTL;
account.setAccountTTL#2442485e ttl:AccountDaysTTL = Bool;

invokeWithLayer#da9b0d0d {X:Type} layer:int query:!X = X;

contacts.resolveUsername#0bf0131c username:string = User;

account.sendChangePhoneCode#a407a8f4 phone_number:string = account.SentChangePhoneCode;
account.changePhone#70c32edb phone_number:string phone_code_hash:string phone_code:string = User;

messages.getStickers#ae22e045 emoticon:string hash:string = messages.Stickers;
messages.getAllStickers#aa3bc868 hash:string = messages.AllStickers;

account.updateDeviceLocked#38df3532 period:int = Bool;
```

End-to-end scheme changes

New Constructors

- Added decryptedMessageActionCommitKey – Commit new key, see rekeying process
- Added decryptedMessageActionNoop – NOOP action
- Added decryptedMessageActionAbortKey – Abort rekeying
- Added decryptedMessageActionAcceptKey – Accept new key
- Added decryptedMessageActionRequestKey – Request rekeying, see rekeying process
- Added documentAttributeImageSize – Defines the width and height of an image uploaded as document
- Added documentAttributeAnimated – Defines an animated GIF
- Added documentAttributeSticker – Defines a sticker
- Added documentAttributeVideo – Defines a video
- Added documentAttributeAudio – Represents an audio file
- Added documentAttributeFilename – A simple document with a file name
- Added photoSizeEmpty – Empty constructor. Image with this thumbnail is unavailable.
- Added photoSize – Image description.
- Added photoCachedSize – Description of an image and its content.
- Added fileLocationUnavailable
- Added fileLocation
- Added decryptedMessageMediaExternalDocument – Non-e2e documented forwarded from non-secret chat

End-to-end scheme

```
===20===

decryptedMessageActionCommitKey#ec2e0b9b exchange_id:long key_fingerprint:long = DecryptedMessageAction;
decryptedMessageActionNoop#a82fdd63 = DecryptedMessageAction;
decryptedMessageActionAbortKey#dd05ec6b exchange_id:long = DecryptedMessageAction;
decryptedMessageActionAcceptKey#6fe1735b exchange_id:long g_b:bytes key_fingerprint:long = DecryptedMessageActi
decryptedMessageActionRequestKey#f3c9611b exchange_id:long g_a:bytes = DecryptedMessageAction;

===23===
documentAttributeImageSize#6c37c15c w:int h:int = DocumentAttribute;
documentAttributeAnimated#11b58939 = DocumentAttribute;
documentAttributeSticker#fb0a5727 = DocumentAttribute;
documentAttributeVideo#5910cccb duration:int w:int h:int = DocumentAttribute;
documentAttributeAudio#051448e5 duration:int = DocumentAttribute;
documentAttributeFilename#15590068 file_name:string = DocumentAttribute;

photoSizeEmpty#0e17e23c type:string = PhotoSize;
photoSize#77bfb61b type:string location:FileLocation w:int h:int size:int = PhotoSize;
photoCachedSize#e9a734fa type:string location:FileLocation w:int h:int bytes:bytes = PhotoSize;

fileLocationUnavailable#7c596b46 volume_id:long local_id:int secret:long = FileLocation;
fileLocation#53d69076 dc_id:int volume_id:long local_id:int secret:long = FileLocation;

decryptedMessageMediaExternalDocument#fa95b0dd id:long access_hash:long date:int mime_type:string size:int thum
```

Layer 18

Added username support and a new type of updates for service messages.

Schema changes

- Added methods account.checkUsername and account.updateUsername for setting up a username, as well as the method contacts.search that allows searching for publicly available users by username.
- **username** field added to constructors of the type User and to the updateUserName constructor.
- Added new update constructor updateServiceNotification for receiving service messages.

Schema

```
contactFound user_id:int = ContactFound;

contacts.found results:Vector<ContactFound> users:Vector<User> = contacts.Found;

updateUserName user_id:int first_name:string last_name:string username:string = Update;

userSelf id:int first_name:string last_name:string username:string phone:string photo:UserProfilePhoto status:U
userContact id:int first_name:string last_name:string username:string access_hash:long phone:string photo:UserP
userRequest id:int first_name:string last_name:string username:string access_hash:long phone:string photo:UserP
userForeign id:int first_name:string last_name:string username:string access_hash:long photo:UserProfilePhoto s
userDeleted id:int first_name:string last_name:string username:string = User;

updateServiceNotification type:string message:string media:MessageMedia popup:Bool = Update;

---functions---
```

```
account.checkUsername username:string = Bool;
account.updateUsername username:string = User;

contacts.search q:string limit:int = contacts.Found;

invokeWithLayer18#1c900537 {X:Type} query:!X = X;
```

Layer 17

Added new events for recording and uploading media, selecting contacts and locations to share.

Read status for multimedia (messages containing messageMediaVideo or messageMediaAudio) was moved to the new method messages.readMessageContents. In case **read_contents** is not passed or messages.readHistory is used with an older layer, messages will be marked as read in the traditional way.

Schema changes

- Added new type SendMessageAction for user actions aside from typing (recording, uploading media, etc.). It is used in updates updateUserTyping, updateChatUserTyping, in the method messages.setTyping, and in the new encrypted service message decryptedMessageActionTyping.
- **unread** and **out** parameters in the constructors of the Message type were joined to form the new **flags** parameter, containing a flag mask.
- Added new method messages.readMessageContents, to be called once the user listened to a voice message or watched a video.
- Added parameters for end-to-end encrypted messages **in_seq_no**, **out_seq_no** and **ttl**.
- Added **mime_type** field to secret chat audio and video constructors.

Schema

```
sendMessageTypingAction = SendMessageAction;
sendMessageCancelAction = SendMessageAction;
sendMessageRecordVideoAction = SendMessageAction;
sendMessageUploadVideoAction = SendMessageAction;
sendMessageRecordAudioAction = SendMessageAction;
sendMessageUploadAudioAction = SendMessageAction;
sendMessageUploadPhotoAction = SendMessageAction;
sendMessageUploadDocumentAction = SendMessageAction;
sendMessageGeoLocationAction = SendMessageAction;
sendMessageChooseContactAction = SendMessageAction;

updateUserTyping user_id:int action:SendMessageAction = Update;
updateChatUserTyping chat_id:int user_id:int action:SendMessageAction = Update;

// Message object
message flags:int id:int from_id:int to_id:Peer date:int message:string media:MessageMedia = Message;
messageForwarded flags:int id:int fwd_from_id:int fwd_date:int from_id:int to_id:Peer date:int message:string m
messageService flags:int id:int from_id:int to_id:Peer date:int action:MessageAction = Message;

---functions---

messages.setTyping peer:InputPeer action:SendMessageAction = Bool;

messages.readHistory peer:InputPeer max_id:int offset:int read_contents:Bool = messages.AffectedHistory;
messages.readMessageContents id:Vector<int> = Vector<int>;

invokeWithLayer17#50858a19 {X:Type} query:!X = X;
```

End-to-end schema

```
===17===


// Layer
decryptedMessageLayer random_bytes:bytes layer:int in_seq_no:int out_seq_no:int message:DecryptedMessage = Decr

decryptedMessageMediaAudio duration:int mime_type:string size:int key:bytes iv:bytes = DecryptedMessageMedia;
decryptedMessageMediaVideo thumb:bytes thumb_w:int thumb_h:int duration:int mime_type:string w:int h:int size:i

sendMessageTypingAction = SendMessageAction;
sendMessageCancelAction = SendMessageAction;
sendMessageRecordVideoAction = SendMessageAction;
sendMessageUploadVideoAction = SendMessageAction;
sendMessageRecordAudioAction = SendMessageAction;
sendMessageUploadAudioAction = SendMessageAction;
sendMessageUploadPhotoAction = SendMessageAction;
sendMessageUploadDocumentAction = SendMessageAction;
sendMessageGeoLocationAction = SendMessageAction;
sendMessageChooseContactAction = SendMessageAction;

decryptedMessageActionNotifyLayer layer:int = DecryptedMessageAction;
decryptedMessageActionTyping action:SendMessageAction = DecryptedMessageAction;

decryptedMessage random_id:long ttl:int message:string media:DecryptedMessageMedia = DecryptedMessage;
decryptedMessageService random_id:long action:DecryptedMessageAction = DecryptedMessage;
```

Layer 16

Added new **sms_type** = `5` in the method auth.sendCode.

Schema changes

- Added new constructor: auth.sentAppCode to determine whether a code was sent via Telegram.
- Added new method auth.sendSms to force re-sending a code via SMS.

Schema

```
auth.sentAppCode phone_registered:Bool phone_code_hash:string send_call_timeout:int is_password:Bool = auth.Sen

---functions---

auth.sendSms phone_number:string phone_code_hash:string = Bool;

invokeWithLayer16#cf5f0987 {X:Type} query:!X = X;
```

Layer 15

Modified behavior of the **offset** parameter in the method messages.getHistory. From now on it's possible to combine message_id offset and a numeric offset.

Schema

```
invokeWithLayer15#b4418b64 {X:Type} query:!X = X;
```

Layer 14

Schema changes

- Added new update constructors: updateUserBlocked and updateNotifySettings to sync notification settings and

blacklists between devices.

- Added **notify_settings** field to the sending constructor.

Schema

```
notifyPeer peer:Peer = NotifyPeer;
notifyUsers = NotifyPeer;
notifyChats = NotifyPeer;
notifyAll = NotifyPeer;

dialog peer:Peer top_message:int unread_count:int notify_settings:PeerNotifySettings = Dialog;

updateUserBlocked user_id:int blocked:Bool = Update;
updateNotifySettings peer:NotifyPeer notify_settings:PeerNotifySettings = Update;

---functions---

invokeWithLayer14#2b9b08fa {X:Type} query:!X = X;
```

Layer 13

Schema changes

- Added **mime_type** field to all audio and video constructors.
- Added new service message types in secret chats: messages read, messages deleted, screenshot taken, chat history cleared and API layer used by client notifications.
- Added **retry_contacts** field to the contacts.importedContacts constructor: ids of contacts, that will have to be imported at a later date.

Schema

```
contacts.importedContacts imported:Vector<ImportedContact> retry_contacts:Vector<long> users:Vector<User> = con

inputMediaUploadedAudio file:InputFile duration:int mime_type:string = InputMedia;
inputMediaUploadedVideo file:InputFile duration:int w:int h:int mime_type:string = InputMedia;
inputMediaUploadedThumbVideo file:InputFile thumb:InputFile duration:int w:int h:int mime_type:string = InputMe

audio id:long access_hash:long user_id:int date:int duration:int mime_type:string size:int dc_id:int = Audio;
video id:long access_hash:long user_id:int date:int caption:string duration:int mime_type:string size:int thumb

---functions---

invokeWithLayer13#427c8ea2 {X:Type} query:!X = X;
```

Layer 12

Schema changes

- Added method help.getSupport for obtaining support user id.
- Added **broadcast_size_max** field to the constructor config, contaning maximum number of broadcast recipients.
- Added **send_call_timeout** field to the constructor auth.sentCode, containing required delay before calling auth.sendCall. New field **is_password** in the same constructor.

Schema

```
auth.sentCode phone_registered:Bool phone_code_hash:string send_call_timeout:int is_password:Bool = auth.SentCo

config date:int test_mode:Bool this_dc:int dc_options:Vector<DcOption> chat_size_max:int broadcast_size_max:int

help.support phone_number:string user:User = help.Support;

---functions---

help.getSupport = help.Support;

invokeWithLayer12#dda60d3c {X:Type} query:!X = X;
```

Layer 11

Schema changes

- The **nonce** parameter was removed from secret chat constructors. For purposes of backward compatibility, in all previous layers this field will contain empty bytes.

Schema

```
encryptedChatRequested id:int access_hash:long date:int admin_id:int participant_id:int g_a:bytes = EncryptedCh
encryptedChat id:int access_hash:long date:int admin_id:int participant_id:int g_a_or_b:bytes key_fingerprint:l

---functions---

invokeWithLayer11#a6b88fdf {X:Type} query:!X = X;
```

Layer 10

Added brief constructors for editing group members. Added new attachment types for ordinary and secret chats.

Schema changes

- Added new update contructors: updateChatParticipantAdd and updateChatParticipantDelete.
- Added new media types: Document, Audio, along with rescpective attachment constructors.
- If the client doesn't support Layer 10, it will receive messageMediaUnsupported constructors instead of the messageMediaDocument and messageMediaAudio constructors.

Schema

```
updateChatParticipantAdd chat_id:int user_id:int inviter_id:int version:int = Update;
updateChatParticipantDelete chat_id:int user_id:int version:int = Update;


inputMediaUploadedAudio file:InputFile duration:int = InputMedia;
inputMediaAudio id:InputAudio = InputMedia;

inputMediaUploadedDocument file:InputFile file_name:string mime_type:string = InputMedia;
inputMediaUploadedThumbDocument file:InputFile thumb:InputFile file_name:string mime_type:string = InputMedia;
inputMediaDocument id:InputDocument = InputMedia;

messageMediaDocument document:Document = MessageMedia;
messageMediaAudio audio:Audio = MessageMedia;

// Input Audio
inputAudioEmpty = InputAudio;
inputAudio id:long access_hash:long = InputAudio;

// Input Document
inputDocumentEmpty = InputDocument;
inputDocument id:long access_hash:long = InputDocument;

// Input location
```

```
inputAudioFileLocation id:long access_hash:long = InputFileLocation;
inputDocumentFileLocation id:long access_hash:long = InputFileLocation;


decryptedMessageMediaDocument thumb:bytes thumb_w:int thumb_h:int file_name:string mime_type:string size:int ke
decryptedMessageMediaAudio duration:int size:int key:bytes iv:bytes = DecryptedMessageMedia;


// Audio object
audioEmpty id:long = Audio;
audio id:long access_hash:long user_id:int date:int duration:int size:int dc_id:int = Audio;

// Video object
documentEmpty id:long = Document;
document id:long access_hash:long user_id:int date:int file_name:string mime_type:string size:int thumb:PhotoSi

---functions---

invokeWithLayer10#39620c41 {X:Type} query:!X = X;
PUSH-notifications

Added MESSAGE_DOC, MESSAGE_AUDIO notifications — a message with document or audio received
Added CHAT_MESSAGE_DOC, CHAT_MESSAGE_AUDIO notifications —  a message with document or audio received in a grou
```

Layer 9

Increased efficiency when loading big files. Added important checks for certain methods.

Schema changes

- Added new upload.saveBigFilePart method, constructors inputFileBig, inputEncryptedFileBigUploaded to optimize saving parts of big files. See the updated article on Uploading files for details.
- Added a check whether the initConnection method was called with all required parameters before making other API calls. See the updated article on Calling methods for details.
- Added availability check for **lang_code** parameters when calling methods auth.sendCode, account.registerDevice
- Added availability check for **random_id** parameters in all message sending methods: messages.sendMessage, messages.sendMedia, messages.forwardMessage

Schema

```
inputFileBig id:long parts:int name:string = InputFile;

inputEncryptedFileBigUploaded id:long parts:int key_fingerprint:int = InputEncryptedFile;

---functions---

upload.saveBigFilePart file_id:long file_part:int file_total_parts:int bytes:bytes = Bool;

initConnection#69796de9 {X:Type} api_id:int device_model:string system_version:string app_version:string lang_c

invokeWithLayer9#76715a63 {X:Type} query:!X = X;
```

Layer 8

Added support for end-to-end encryption in secret chats. More...

Schema changes

- Added many constructors and methods for secret chats.
- Added **qts** field in constructor updates.state.
- Added 4 new constructors of Update type

Schema

```
updateNewEncryptedMessage message:EncryptedMessage qts:int = Update;
updateEncryptedChatTyping chat_id:int = Update;
updateEncryption chat:EncryptedChat date:int = Update;
updateEncryptedMessagesRead chat_id:int max_date:int = Update;

// EncryptedChat object
encryptedChatEmpty id:int = EncryptedChat;
encryptedChatWaiting id:int access_hash:long date:int admin_id:int participant_id:int = EncryptedChat;
encryptedChatRequested id:int access_hash:long date:int admin_id:int participant_id:int g_a:bytes = EncryptedCh
encryptedChat id:int access_hash:long date:int admin_id:int participant_id:int g_a_or_b:bytes key_fingerprint:l
encryptedChatDiscarded id:int = EncryptedChat;

inputEncryptedChat chat_id:int access_hash:long = InputEncryptedChat;

// EncryptedFile object
encryptedFileEmpty = EncryptedFile;
encryptedFile id:long access_hash:long size:int dc_id:int key_fingerprint:int = EncryptedFile;

inputEncryptedFileEmpty = InputEncryptedFile;
inputEncryptedFileUploaded id:long parts:int md5_checksum:string key_fingerprint:int = InputEncryptedFile;
inputEncryptedFile id:long access_hash:long = InputEncryptedFile;

inputEncryptedFileLocation id:long access_hash:long = InputFileLocation;

// Encrypted message
encryptedMessage random_id:long chat_id:int date:int bytes:bytes file:EncryptedFile = EncryptedMessage;
encryptedMessageService random_id:long chat_id:int date:int bytes:bytes = EncryptedMessage;

// Diffie-Hellman config
messages.dhConfigNotModified random:bytes = messages.DhConfig;
messages.dhConfig g:int p:bytes version:int random:bytes = messages.DhConfig;

messages.sentEncryptedMessage date:int = messages.SentEncryptedMessage;
messages.sentEncryptedFile date:int file:EncryptedFile = messages.SentEncryptedMessage;

// Updated state with qts
updates.state pts:int qts:int date:int seq:int unread_count:int = updates.State;
updates.difference new_messages:Vector<Message> new_encrypted_messages:Vector<EncryptedMessage> other_updates:V
updates.differenceSlice new_messages:Vector<Message> new_encrypted_messages:Vector<EncryptedMessage> other_upda

---functions---

messages.getDhConfig version:int random_length:int = messages.DhConfig;
messages.requestEncryption user_id:InputUser random_id:int g_a:bytes = EncryptedChat;
messages.acceptEncryption peer:InputEncryptedChat g_b:bytes key_fingerprint:long = EncryptedChat;
messages.discardEncryption chat_id:int = Bool;

messages.setEncryptedTyping peer:InputEncryptedChat typing:Bool = Bool;
messages.readEncryptedHistory peer:InputEncryptedChat max_date:int = Bool;
messages.sendEncrypted peer:InputEncryptedChat random_id:long data:bytes = messages.SentEncryptedMessage;
messages.sendEncryptedFile peer:InputEncryptedChat random_id:long data:bytes file:InputEncryptedFile = messages
messages.sendEncryptedService peer:InputEncryptedChat random_id:long data:bytes = messages.SentEncryptedMessage
messages.receivedQueue max_qts:int = Vector<long>;

updates.getDifference pts:int date:int qts:int = updates.Difference;

invokeWithLayer8#e9abd9fd {X:Type} query:!X = X;
```

Push-notifications

- Added notification of ENCRYPTION_REQUEST type -- a contact requested secret chat creation
- Added notification of ENCRYPTION_ACCEPT type -- a contact confirmed secret chat creation
- Added notification of ENCRYPTED_MESSAGE type -- a contact sent message in a secret chat

End-to-end schema

```
===8===
decryptedMessage random_id:long random_bytes:bytes message:string media:DecryptedMessageMedia = DecryptedMessag
decryptedMessageService random_id:long random_bytes:bytes action:DecryptedMessageAction = DecryptedMessage;

decryptedMessageMediaEmpty = DecryptedMessageMedia;
decryptedMessageMediaPhoto thumb:bytes thumb_w:int thumb_h:int w:int h:int size:int key:bytes iv:bytes = Decryp
decryptedMessageMediaVideo thumb:bytes thumb_w:int thumb_h:int duration:int w:int h:int size:int key:bytes iv:b
decryptedMessageMediaGeoPoint lat:double long:double = DecryptedMessageMedia;
decryptedMessageMediaContact phone_number:string first_name:string last_name:string user_id:int = DecryptedMess

decryptedMessageActionSetMessageTTL ttl_seconds:int = DecryptedMessageAction;

decryptedMessageMediaDocument thumb:bytes thumb_w:int thumb_h:int file_name:string mime_type:string size:int ke
decryptedMessageMediaAudio duration:int size:int key:bytes iv:bytes = DecryptedMessageMedia;

decryptedMessageActionReadMessages random_ids:Vector<long> = DecryptedMessageAction;
decryptedMessageActionDeleteMessages random_ids:Vector<long> = DecryptedMessageAction;
decryptedMessageActionScreenshotMessages random_ids:Vector<long> = DecryptedMessageAction;
decryptedMessageActionFlushHistory = DecryptedMessageAction;
```

Layer 7

Added wallpaper constructor wallPaperSolid.

Schema

```
wallPaperSolid id:int title:string bg_color:int color:int = WallPaper;
---functions---
invokeWithLayer7#a5be56d3 {X:Type} query:!X = X;
```

Layer 6

Added location identifiers from foursquare.

Schema

```
geoChat id:int access_hash:long title:string address:string venue:string geo:GeoPoint photo:ChatPhoto participa
---functions---
geochats.createGeoChat title:string geo_point:InputGeoPoint address:string venue:string = geochats.StatedMessag

invokeWithLayer6#3a64d54d {X:Type} query:!X = X;
```

Layer 5

Added parameters for internationalization.

Schema changes

- Added parameter lang_code to methods auth.sendCode, account.registerDevice

Schema

```
---functions---
auth.sendCode phone_number:string sms_type:int api_id:int api_hash:string lang_code:string = auth.SentCode;
account.registerDevice token_type:int token:string device_model:string system_version:string app_version:string

invokeWithLayer5#417a57ae {X:Type} query:!X = X;
```

Layer 4

Added geochats. More...

Schema changes

- Added many constructors and methods for geochats.
- Added friends_unread_count field to constructor updates.state.

Schema

```
inputGeoChat chat_id:int access_hash:long = InputGeoChat;

geoChat id:int access_hash:long title:string address:string geo:GeoPoint photo:ChatPhoto participants_count:int

geoChatMessageEmpty chat_id:int id:int = GeoChatMessage;
geoChatMessage chat_id:int id:int from_id:int date:int message:string media:MessageMedia = GeoChatMessage;
geoChatMessageService chat_id:int id:int from_id:int date:int action:MessageAction = GeoChatMessage;

geochats.statedMessage message:GeoChatMessage chats:Vector<Chat> users:Vector<User> seq:int = geochats.StatedMe

geochats.located results:Vector<ChatLocated> messages:Vector<GeoChatMessage> chats:Vector<Chat> users:Vector<Us

geochats.messages messages:Vector<GeoChatMessage> chats:Vector<Chat> users:Vector<User> = geochats.Messages;
geochats.messagesSlice count:int messages:Vector<GeoChatMessage> chats:Vector<Chat> users:Vector<User> = geocha

messageActionGeoChatCreate title:string address:string = MessageAction;
messageActionGeoChatCheckin = MessageAction;
updateNewGeoChatMessage message:GeoChatMessage = Update;
updates.state pts:int date:int seq:int unread_count:int friends_unread_count:int = updates.State;

---functions---

geochats.getLocated geo_point:InputGeoPoint radius:int limit:int = geochats.Located;
geochats.checkin peer:InputGeoChat = geochats.StatedMessage;
geochats.getFullChat peer:InputGeoChat = messages.ChatFull;
geochats.editChatTitle peer:InputGeoChat title:string address:string = geochats.StatedMessage;
geochats.editChatPhoto peer:InputGeoChat photo:InputChatPhoto = geochats.StatedMessage;
geochats.search peer:InputGeoChat q:string filter:MessagesFilter min_date:int max_date:int offset:int max_id:in
geochats.getHistory peer:InputGeoChat offset:int max_id:int limit:int = geochats.Messages;
geochats.setTyping peer:InputGeoChat typing:Bool = Bool;
geochats.sendMessage peer:InputGeoChat message:string random_id:long = geochats.StatedMessage;
geochats.sendMedia peer:InputGeoChat media:InputMedia random_id:long = geochats.StatedMessage;
geochats.createGeoChat title:string geo_point:InputGeoPoint address:string = geochats.StatedMessage;

invokeWithLayer4#dea0d430 {X:Type} query:!X = X;
```

Push-notifications

- Added notification of GEOCHAT_CHECKIN type -- a user has checked-in in a geochat.

Layer 3

Support for link changes for a contact when a message is sent. Now, if user X has user Y in the contact list and if user Y writes a message to user X, number X will become available for him. More...

Schema changes

- Added constructors messages.statedMessagesLinks, messages.statedMessageLink, messages.sentMessageLink. They are completely similar to previous ones except for the links field containing changed links.
- Added events_mask field to constructors inputPeerNotifySettings, peerNotifySettings making possible turning-off PUSH-notifications on image changes.
- Added method messages.forwardMessage for sending single messages.
- Added method messages.sendBroadcast for bulky messaging.

Schema

```
messages.statedMessagesLinks messages:Vector<Message> chats:Vector<Chat> users:Vector<User> links:Vector<contac
messages.statedMessageLink message:Message chats:Vector<Chat> users:Vector<User> links:Vector<contacts.Link> pt
messages.sentMessageLink id:int date:int pts:int seq:int links:Vector<contacts.Link> = messages.SentMessage;

---functions---

messages.forwardMessage peer:InputPeer id:int random_id:long = messages.StatedMessage;
messages.sendBroadcast contacts:Vector<InputUser> message:string media:InputMedia = messages.StatedMessages;

invokeWithLayer3#b7475268 {X:Type} query:!X = X;
```

Layer 2

Support for notifications on changes of contact profile images. It is assumed that receiving such image changed notification a client will add a record on this event in the message history with this contact.

Schema changes

- Added date and previous fields to constructor updateUserPhoto.
- Added events_mask field to constructors inputPeerNotifySettings, peerNotifySettings making possible turning-off PUSH-notifications on image change.
- Added identifier for relevant image to constructor userProfilePhoto.
- Added method photos.getUserPhotos to get a history of previous profile images.

Schema

```
inputPeerNotifySettings mute_until:int sound:string show_previews:Bool events_mask:int = InputPeerNotifySetting
peerNotifySettings mute_until:int sound:string show_previews:Bool events_mask:int = PeerNotifySettings;
updateUserPhoto user_id:int date:int photo:UserProfilePhoto previous:Bool = Update;
userProfilePhoto photo_id:long photo_small:FileLocation photo_big:FileLocation = UserProfilePhoto;

---functions---

photos.getUserPhotos user_id:InputUser offset:int max_id:int limit:int = photos.Photos;

invokeWithLayer2#289dd1f6 {X:Type} query:!X = X;
```

Push-notifications

- Added notification of CONTACT_PHOTO type -- a contact has changed profile image.

Admin, banned, default rights

Channels and supergroups allow setting granular permissions both for admins and specific users.
Channels, supergroups and legacy groups also allow setting global granular permissions for users.

They can be modified as follows:

Admin rights

channels.editAdmin can be used to modify the admin rights of a user in a channel or supergroup.
Legacy groups do not allow setting granular admin permissions, messages.editChatAdmin has to be used, instead.

Permissions are defined by the chatAdminRights constructor, some admin rights can only be used for channels, others both for channels and supergroups (see the constructor page).

Banned rights

channels.editBanned can be used to modify the rights of a user in a channel or supergroup, to ban/kick a user from the group, or restrict the user from doing certain things.
Legacy groups do not allow setting granular user permissions for single users, single users can only be removed from groups using messages.deleteChatUser: however, setting global granular permissions with legacy groups is supported.

Permissions are defined by the chatBannedRights constructor, for more info see the constructor page.

Default rights

messages.editChatDefaultBannedRights can be used to modify the rights of **all** users in a channel, supergroup or legacy group, to restrict them from doing certain things.

Permissions are defined by the chatBannedRights constructor: all flags can be used except for `view_messages`, for more info see the constructor page.

Min constructors

In some situations user and channel constructors have reduced set of fields present (although `id` is always there) and `min` flag set. This is done for performance and privacy reasons.

When receiving said constructors, the client must first check if user or chat object without `min` flag is already present in local cache. If it is present, then the client should just ignore constructors with `min` flag and use local one instead.

The rest of article assumes the client receives min-constructor without full object in local cache.

The client must store the context (similar to file references) in which the user/channel was seen. Later, when the client needs to pass the user/channel as input argument (e.g. fetch profile, mute, ban etc), the context is used to generate the `input*FromMessage` constructor, instead of normal `inputUser`, `inputChannel` or `inputPeer`.

- inputPeerUserFromMessage
- inputPeerChannelFromMessage
- inputUserFromMessage
- inputChannelFromMessage

The `access_hash` value, if present, is only suitable to use in `inputPeerPhotoFileLocation`, to directly download the profile pictures of channels and users **without** having to generate an `inputPeer*FromMessage`, simply using `inputPeer*` with the specified access hash.

Usually `min` constructors are encountered in messages inside of groups or channels.
When a message mentioning (sender, forwarder or forwardee, et cetera) such a user or channel is found, the constuctor must be associated with the message ID of the message and with the chat where the message was seen.

Example

Assume a message with id `34` is received from supergroup (actually channel) `123456789`.
Said message was sent by `from_id` `102424212`.
The updates container that contained the message has a user with ID `102424212` in the `users` field, but it has the `min` flag set, and has no `access_hash` (thus it can't be used to generate a typical inputPeerUser constructor to send messages or do other actions).

What the client does is associate `102424212` with the channel `123456789` and message ID `34`.
When and if the client will need to interact with user `102424212`, it will generate one of the `*FromMessage` constructors mentioned above, setting:

- `msg_id` to `34`
- `peer` to the InputPeer associated with channel `123456789`
- `user_id` to `102424212`

`user_id` can also be set to the IDs of users met in the `fwd_header` (messages forwarded from a user can be used to interact with the original sender, if he doesn't have privacy settings for forwards enabled).
Users mentioned via messageEntityMentionName in a message can also be used.

The same can be done with `min` channels.

Example implementations: Telegram for iOS, tdlib.

Account deletion

If the user has successfully provided the login code, but they forgot the 2FA password, the account should be reset: this can be done using account.deleteAccount.

If the account's 2FA password was modified more than 7 days ago and was active in the last 7 days, account deletion will be delayed for 7 days, and a service message will be sent to the user, containing a link in one of the following formats:

- `https://telegram.me/confirmphone?phone=XXX&hash=YYYY`
- `tg://confirmphone?phone=XXX&hash=YYYY`

When clicked, account.sendConfirmPhoneCode must be called with the specified `hash`, using the account with the specified `phone` number.
This will send a phone number verification code to the phone number associated with the account.
The phone code settings are the same as for the login code, and auth.cancelCode with auth.resendCode can be used as well, to resend or cancel the phone code as for the login code.

Once the SMS code is received, the account.confirmPhone method will have to be called with the SMS code and the phone hash received from the account.sendConfirmPhoneCode method.

This will cancel deletion of the account and will log out the user that tried to reset it.
Otherwise, if the number isn't confirmed in 7 days, the account will be deleted and the user will be free to recreate it.

Related articles

User Authorization

How to register a user's phone to start using the API.

Telegram passport

Telegram Passport is a unified authorization method for services that require personal identification. Users can upload their documents once, then instantly share their data with services that require real-world ID (finance, ICOs, etc.). Telegram doesn't have access to the users' personal information thanks to end-to-end encryption.

This page describes the request flow that client apps must used to send the requested data to the service.

Overview

From the perspective of a service that requires real-world ID, the process looks like this:

- A user presses "Log in with Telegram" on your website or in your app.
- You request the data you need.
- The user accepts your privacy policy and agrees to share their data.
- The user's Telegram app downloads and decrypts the data you requested from the end-to-end encrypted storage on Telegram.
- If some of the data you requested is missing, the user can add it to their Telegram Passport at this point.
- The user's app encrypts the data with your public key and sends it to you.
- You decrypt the data, check it for errors and re-request any missing or invalid information.
- You sign the user up for your service. Tada!

See As a bot to see how to request passport data using a bot, through the MTProto API.
Look at the Passport Manual to see how to request passport data using a bot, through the simplified bot API.

From the perspective of a user, the process looks something like this:

- Your app receives an event/intent from one of the SDKs, or from a custom source.
- The user accepts your privacy policy and agrees to share their data.
- The user's Telegram app downloads the data you requested from the end-to-end encrypted storage on Telegram.
- If some of the data you requested is missing, the user can add it to their Telegram Passport at this point.
- The user's app encrypts the data with your public key and sends it to the service.
- You sign the user up for your service. Tada!

See As a user to see how user client apps should send passport data to a service, through the MTProto API.

As a bot

A simplified version of this process can be used using the bot API, for more info see the Passport Manual.

Using the MTProto API, the process is pretty much the same, up until the actual API calls.

> Note that all binary fields are in raw binary format, unlike in the bot API where they are base64-encoded

Setting Up Telegram Passport

As per the bot API.

Requesting Information

As per the bot API.

Receiving information

Scheme:

```
secureData#8aeabec3 data:bytes data_hash:bytes secret:bytes = SecureData;

securePlainPhone#7d6099dd phone:string = SecurePlainData;
securePlainEmail#2/passport/encryption#when-to-use-each-constructor1ec5a5f email:string = SecurePlainData;

secureFile#e0277a62 id:long access_hash:long size:int dc_id:int date:int file_hash:bytes secret:bytes = SecureF

secureValueTypePersonalDetails#9d2a81e3 = SecureValueType;
secureValueTypePassport#3dac6a00 = SecureValueType;
secureValueTypeDriverLicense#6e425c4 = SecureValueType;
secureValueTypeIdentityCard#a0d0744b = SecureValueType;
secureValueTypeInternalPassport#99a48f23 = SecureValueType;
secureValueTypeAddress#cbe31e26 = SecureValueType;
secureValueTypeUtilityBill#fc36954e = SecureValueType;
secureValueTypeBankStatement#89137c0d = SecureValueType;
secureValueTypeRentalAgreement#8b883488 = SecureValueType;
secureValueTypePassportRegistration#99e3806a = SecureValueType;
secureValueTypeTemporaryRegistration#ea02ec33 = SecureValueType;
secureValueTypePhone#b320aadb = SecureValueType;
secureValueTypeEmail#8e3ca7ee = SecureValueType;

secureValue#187fa0ca flags:# type:SecureValueType data:flags.0?SecureData front_side:flags.1?SecureFile reverse

secureCredentialsEncrypted#33f0ea47 data:bytes hash:bytes secret:bytes = SecureCredentialsEncrypted;

messageActionSecureValuesSentMe#1b287353 values:Vector<SecureValue> credentials:SecureCredentialsEncrypted = Me
messageService#9e19a1f6 flags:# out:flags.1?true mentioned:flags.4?true media_unread:flags.5?true silent:flags.

updateNewMessage#1f2b0afd message:Message pts:int pts_count:int = Update;
```

When the user confirms your request by pressing the 'Authorize' button, the MTProto API sends an updateNewMessage from the user, with a messageService constructor, containing a messageActionSecureValuesSentMe constructor that contains the encrypted Telegram Passport data.

Decrypting data

```
secureCredentialsEncrypted#33f0ea47 data:bytes hash:bytes secret:bytes = SecureCredentialsEncrypted;

messageActionSecureValuesSentMe#1b287353 values:Vector<SecureValue> credentials:SecureCredentialsEncrypted = Me
```

To decrypt the received data, first, decrypt the credentials contained in secureCredentialsEncrypted.

1. Decrypt the credentials secret (*secret* field in secureCredentialsEncrypted) using your **private** key (set OAEP padding option, e.g. `OPENSSL_PKCS1_OAEP_PADDING` in PHP)
2. Use this secret and the credentials hash (*hash* field in secureCredentialsEncrypted) to calculate *credentials_key* and *credentials_iv* as described below:

   ```
   credentials_secret_hash = SHA512( credentials_secret + credentials_hash )
   credentials_key = slice( credentials_secret_hash, 0, 32 )
   credentials_iv = slice( credentials_secret_hash, 32, 16 )
   ```

3. Decrypt the credentials data (*data* field in secureCredentialsEncrypted) by AES256-CBC using these *credentials_key* and *credentials_iv*. **IMPORTANT**: At this step, make sure that the credentials hash is equal to `SHA256( credentials_data )`
4. Credentials data is padded with 32 to 255 random padding bytes to make its length divisible by 16 bytes. The first byte contains the length of this padding (including this byte). Remove the padding to get the data.

> Note that all hashes are raw binary data, not hexits

Credentials

The credentials are a JSON-serialized object, structured exactly as in the bot API ».
Since decryption credentials are E2E encrypted, apps have to store the decryption credentials as JSON and not TL payloads.

The credentials are used as described in the Passport Manual to decrypt the files attached to the secureValue.
In this case, the container for the E2E encrypted data is in TL, while the encrypted data itself is in JSON.

secureValue

```
secureValue#187fa0ca flags:# type:SecureValueType data:flags.0?SecureData front_side:flags.1?SecureFile reverse

messageActionSecureValuesSentMe#1b287353 values:Vector<SecureValue> credentials:SecureCredentialsEncrypted = Me
```

The scheme for the secureValue constructor defines the constructor that can be found in each field.

Name	Type	Description
type	SecureValueType	Secure passport value type
data	flags.0?SecureData	Encrypted Telegram Passport element data
front_side	flags.1?SecureFile	Encrypted passport file with the front side of the document
reverse_side	flags.2?SecureFile	Encrypted passport file with the reverse side of the document
selfie	flags.3?SecureFile	Encrypted passport file with a selfie of the user holding the document
translation	flags.6?Vector<SecureFile>	Array of encrypted passport files with translated versions of the provided documents
files	flags.4?Vector<SecureFile>	Array of encrypted passport files with photos the of the documents
plain_data	flags.5?SecurePlainData	Plaintext verified passport data
hash	bytes	Data hash

Here's a list of possible SecureValueTypes, and the parameters that can be set/requested when using each type.

Type	Allowed fields
secureValueTypeEmail	`plain_data`
secureValueTypePhone	`plain_data`
secureValueTypePersonalDetails	`data`
secureValueTypePassport	`data`, `front_side`, `selfie`, `translation`
secureValueTypeDriverLicense	`data`, `front_side`, `reverse_side`, `selfie`, `translation`
secureValueTypeIdentityCard	`data`, `front_side`, `reverse_side`, `selfie`, `translation`
secureValueTypeInternalPassport	`data`, `front_side`, `selfie`, `translation`
secureValueTypeAddress	`data`
secureValueTypeUtilityBill	`files`, `translation`
secureValueTypeBankStatement	`files`, `translation`
secureValueTypeRentalAgreement	`files`, `translation`
secureValueTypePassportRegistration	`files`, `translation`
secureValueTypeTemporaryRegistration	`files`, `translation`

SecureData

```
secureData#8aeabec3 data:bytes data_hash:bytes secret:bytes = SecureData;
```

Data is an encrypted and padded JSON-serialized object of one of the specified JSON types, depending on the chosen type.

Chosen type	JSON object
secureValueTypePersonalDetails	PersonalDetails
secureValueTypePassport	IdDocumentData
secureValueTypeDriverLicense	IdDocumentData
secureValueTypeIdentityCard	IdDocumentData
secureValueTypeInternalPassport	IdDocumentData
secureValueTypeAddress	ResidentialAddress

DataCredentials extracted from the credentials can then be used to decrypt encrypted data from the *data* field in secureData.
For more info on how to decrypt the *data* field, see the passport manual.

SecureFile

```
secureFile#e0277a62 id:long access_hash:long size:int dc_id:int date:int file_hash:bytes secret:bytes = SecureF

inputSecureFileLocation#cbc7ee28 id:long access_hash:long = InputFileLocation;

---functions---

upload.getFile#b15a9afc flags:# precise:flags.0?true location:InputFileLocation offset:int limit:int = upload.F
```

Files (JPG format when decrypted, max. 10MB) are downloaded chunk by chunk as described in files », except that instead of generating an inputFileLocation, an inputFileLocation should be generated, instead.

- The `id` field is the `id` of the secureFile
- The `access_hash` field is the `access_hash` of the secureFile

FileCredentials extracted from the credentials can then be used to decrypt downloaded encrypted data.
For more info on how to decrypt passport files, see the passport manual.

SecurePlainData

```
securePlainPhone#7d6099dd phone:string = SecurePlainData;
securePlainEmail#21ec5a5f email:string = SecurePlainData;
```

The email/phone is passed in plaintext using the respective SecurePlainData constructor.
Emails and phone numbers sent using telegram passport are *already verified* as described in the passport manual.

Fixing errors

```
secureValueErrorDataSa40d9 type:SecureValueType data_hash:bytes field:string text:string = SecureValueError;
secureValueErrorFrontSide#be3dfa type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorReverseSide#868a2aa5 type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorSelfie#e537ced6 type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorFile#7a700873 type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorFiles#666220e9 type:SecureValueType file_hash:Vector<bytes> text:string = SecureValueError;
secureValueError#869d758f type:SecureValueType hash:bytes text:string = SecureValueError;
secureValueErrorTranslationFile#a1144770 type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorTranslationFiles#34636dd8 type:SecureValueType file_hash:Vector<bytes> text:string = SecureValu

inputUser#d8292816 user_id:int access_hash:long = InputUser;


---functions---

users.setSecureValueErrors#90c894b5 id:InputUser errors:Vector<SecureValueError> = Bool;
```

If the data you received contains errors, the bot can use the users.setSecureValueErrors method to inform the user and request information again. The user will not be able to resend the data, until all errors are fixed.

Descriptions of the method parameters can be found in the method's documentation page ».

As a user

Receiving requests

The process starts when your app receives an event from one of the SDKs, or from a custom source.

URI format

The SDKs trigger a passport authorization request by opening the following Telegram-specific URI:

```
tg://resolve?params
```

With the following query string parameters:

Parameters	Type	Required	Description
domain	String	Yes	Always `telegrampassport` for Passport authorization requests. `tg://` URI are also used to resolve usernames, stickersets, translation packs and more, the `domain` parameter identifies the action to take when opening the link.
bot_id	Integer	Yes	Unique identifier for the bot. You can get it from bot token. For example, for the bot token `1234567:4TT8bAc8GHUspu3ERYn-KGcvsvGB9u_n4ddy`, the bot id is `1234567`.
scope	UriPassportScope	Yes	A more compact JSON-serialized object describing the data you want to request
public_key	String	Yes	Public key of the bot
nonce	String	Yes	Bot-specified nonce. **Important**: For security purposes it should be a cryptographically secure unique identifier of the request. In particular, it should be long enough and it should be generated using a cryptographically secure pseudorandom number generator. You should never accept credentials with the same nonce twice.
callback_url	String	Optional	Supported by some Telegram clients, specifies a callback URL to open once the process is finished or canceled.
payload	String	Optional	**Deprecated** parameter from Telegram Passport 1.0 that had the same function of the `nonce` parameter. Services that still use a legacy version of the SDK may provide this parameter instead of the `nonce`. In some cases, both the `nonce` and the `payload` parameters may be found in a URI, for backwards compatibility: in this case, the `nonce` parameter should always be used instead of `payload`.

Example URI, generated by the Telegram Passport Example page:

```
tg://resolve?domain=telegrampassport&bot_id=543260180&scope=%7B%22v%22%3A1%2C%22d%22%3A%5B%7B%22_%22%3A%22pd%22
```

UriPassportScope

This object represents the data to be requested.

Field	Type	Description
d	Array of UriPassportScopeElement	List of requested elements, each type may be used only once in the entire array of UriPassportScopeElement objects
v	Integer	Scope version, must be *1*

UriPassportScopeElement

This object represents a requested element, should be one of:

- UriPassportScopeElementOneOfSeveral – use to request any one of the documents included in the scope.
- UriPassportScopeElementOne – use to request one particular document.

Passport document type identifiers are aliased with the following reduced type identifiers:

Full	Alias
`personal_details`	`pd`
`passport`	`pp`
`driver_license`	`dl`
`identity_card`	`ic`
`internal_passport`	`ip`
`id_document`	`idd`
`address`	`ad`
`utility_bill`	`ub`
`bank_statement`	`bs`
`rental_agreement`	`ra`
`passport_registration`	`pr`
`temporary_registration`	`tr`
`address_document`	`add`
`phone_number`	`pn`

`email` `em`

You can use the special type “idd” as an alias for one of “pp”, “dl”, “ic” and the special type “add” as an alias for one of “ub”, “bs”, “ra”.

UriPassportScopeElementOneOfSeveral

This object represents several elements one of which must be provided.

Field	Type	Description
_	Array of UriPassportScopeElementOne	List of elements one of which must be provided; must contain either several of “pp”, “dl”, “ic”, “ip” **or** several of “ub”, “bs”, “ra”, “pr”, “tr”
s	Boolean	*Optional.* Use this parameter if you want to request a selfie with the document from this list that the user chooses to upload.
t	Boolean	*Optional.* Use this parameter if you want to request a translation of the document from this list that the user chooses to upload. **Note**: We suggest to only request translations *after* you have received a valid document that requires one.

UriPassportScopeElementOne

This object represents one particular element that must be provided. If no options are needed, *String* can be used instead of this object to specify the type of the element.

Field	Type	Description
_	String	Element type. One of “pd”, “pp”, “dl”, “ic”, “ip”, “ad”, “ub”, “bs”, “ra”, “pr”, “tr”, “pn”, “em”
s	Boolean	*Optional.* Use this parameter if you want to request a selfie with the document as well. Available for “pp”, “dl”, “ic” and “ip”
t	Boolean	*Optional.* Use this parameter if you want to request a translation of the document as well. Available for “pp”, “dl”, “ic”, “ip”, “ub”, “bs”, “ra”, “pr” and “tr”. **Note**: We suggest to only request translations *after* you have received a valid document that requires one.
n	Boolean	*Optional.* Use this parameter to request the first, last and middle name of the user in the language of the user's country of residence. Available for “pd”

You can also use the special type “idd” as an alias for one of “pp”, “dl”, “ic” and the special type “add” as an alias for one of “ub”, “bs”, “ra”.

Setting up Telegram Passport

The next step for the client app is to request the user's 2FA passport, and configure Telegram Passport/fetch and decrypt remotely saved Telegram Passport parameters as described in the Encryption article ».

Fetching the passport form

```
account.authorizationForm#ad2e1cd8 flags:# required_types:Vector<SecureRequiredType> values:Vector<SecureValue>

---functions---

account.getAuthorizationForm#b86ba8e1 bot_id:int scope:string public_key:string = account.AuthorizationForm;
```

Then, the client app passes the bot ID, scope and public key from the passport authorization request to the Telegram servers using the account.getAuthorizationForm method.

The response will be an account.authorizationForm constructor, with info about the required document types, the URL of the service's privacy policy, as well as info about the bot to which the form should be sent.
If the form was already submitted at least once, the constructor will also contain a list of already submitted data, along with eventual errors.

The user should accept the privacy policy and proceed to fill in the required data, and the client should encrypt and upload it as described in the Encryption article ».

Submitting the passport form

```
secureCredentialsEncrypted#33f0ea47 data:bytes hash:bytes secret:bytes = SecureCredentialsEncrypted;

secureValueHash#ed1ecdb0 type:SecureValueType hash:bytes = SecureValueHash;

---functions---

account.acceptAuthorization#e7027c94 bot_id:int scope:string public_key:string value_hashes:Vector<SecureValueH
```

Once the user finishes uploading the required documents and clicks on the submit button, the client calls account.acceptAuthorization, submitting the documents to the bot associated to the service.

- As before, `bot_id`, `scope` and `public_key` are taken from the authorization request URI.
- `value_hashes` is used by the server to choose which document of which type to send to the bot: the `type` field should be set to the document type, and the `hash` field should be set to the `data_hash` / `file_hash` generated when uploading encrypting the data ».
- `credentials` contains the encrypted credentials required by the service to decrypt the sent E2E encrypted secure values: it is generated as described in Passport Credentials ».

Finally, the client opens the callback URL (if present).

Handling invalid forms

```
secureValueErrorData#e8a40bd9 type:SecureValueType data_hash:bytes field:string text:string = SecureValueError;
secureValueErrorFrontSide#be3dfa type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorReverseSide#868a2aa5 type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorSelfie#e537ced6 type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorFile#7a700873 type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorFiles#666220e9 type:SecureValueType file_hash:Vector<bytes> text:string = SecureValueError;
secureValueError#869d758f type:SecureValueType hash:bytes text:string = SecureValueError;
secureValueErrorTranslationFile#a1144770 type:SecureValueType file_hash:bytes text:string = SecureValueError;
secureValueErrorTranslationFiles#34636dd8 type:SecureValueType file_hash:Vector<bytes> text:string = SecureValu

account.authorizationForm#ad2e1cd8 flags:# required_types:Vector<SecureRequiredType> values:Vector<SecureValue>

---functions---

account.getAuthorizationForm#b86ba8e1 bot_id:int scope:string public_key:string = account.AuthorizationForm;
```

If any of the values of the submitted form are rejected by the service, the bot calls the appropriate method to set information about errors.

The user can find out about these errors directly from the service, or, if he decides to restart the process and resend corrected data, directly from the authorization form (`errors` field).

Payments API

You can accept payments from Telegram users via Telegram Bots.

> Note: This article is intended for MTProto API developers. If you're looking for a general overview of Telegram Payments, check out the Telegram blog and the bot API payment manual.

Introducing Payments

Telegram bots can accept payments for goods and services from users.
For more info on how payments work, check out the Telegram Blog and the bot API payment manual.

This page will elaborate on the actions required to work with payments using the **MTProto API**.

> A simplified version of the process is available only for bots using the bot API.

The first step for bots is enable payments as described here ».

Then, we work with payments as follows.

1. Create Invoice

```
inputWebDocument#9bed434d url:string size:int mime_type:string attributes:Vector<DocumentAttribute> = InputWebD

labeledPrice#cb296bf8 label:string amount:long = LabeledPrice;

invoice#c30aa358 flags:# test:flags.0?true name_requested:flags.1?true phone_requested:flags.2?true email_reque

inputMediaInvoice#f4e096c3 flags:# title:string description:string photo:flags.0?InputWebDocument invoice:Invoi

---functions---

messages.sendMedia#b8d1262b flags:# silent:flags.5?true background:flags.6?true clear_draft:flags.7?true peer:I
```

The user contacts the bot and requests to purchase something.
The bot forms an inputMediaInvoice with an invoice constructor with a description of the goods or service, amount to be paid, as well as requested shipping info.
The `provider` parameter of the inputMediaInvoice constructor is where you put the token value that you've obtained earlier via Botfather. It is possible for one merchant bot to use several different tokens for different users or different goods and services.

Use the messages.sendMedia method to send the invoice.
You can also attach an inline keyboard to the message using the `reply_markup` field: if provided, the first button must be a keyboardButtonBuy button. Otherwise, an inline keyboard will be generated automatically, with a `Pay 'total price'` keyboardButtonBuy as only button.

An invoice message with a pay button can only be sent to a private chat with the user. Groups and channels are not supported.

2. Order information

2.1 Invoice message

```
keyboardButtonBuy#afd93fbb text:string = KeyboardButton;

keyboardButtonRow#77608b83 buttons:Vector<KeyboardButton> = KeyboardButtonRow;
replyInlineMarkup#48a30254 rows:Vector<KeyboardButtonRow> = ReplyMarkup;

webDocument#1c570ed1 url:string access_hash:long size:int mime_type:string attributes:Vector<DocumentAttribute>
webDocumentNoProxy#f9c8bcc6 url:string size:int mime_type:string attributes:Vector<DocumentAttribute> = WebDocu

messageMediaInvoice#84551347 flags:# shipping_address_requested:flags.1?true test:flags.3?true title:string des

message#44f9b43d flags:# out:flags.1?true mentioned:flags.4?true media_unread:flags.5?true silent:flags.13?true

updateNewMessage#1f2b0afd message:Message pts:int pts_count:int = Update;
```

The user receives an updateNewMessage constructor from the bot, containing a messageMediaInvoice constructor with basic info about the product.

The message will also have a replyInlineMarkup keyboard attached to it.
The the first button of the keyboard will always be a keyboardButtonBuy button.

2.2 Getting invoice info about the product

```
invoice#c30aa358 flags:# test:flags.0?true name_requested:flags.1?true phone_requested:flags.2?true email_reque

paymentRequestedInfo#909c3f94 flags:# name:flags.0?string phone:flags.1?string email:flags.2?string shipping_ad

paymentSavedCredentialsCard#cdc27a1f id:string title:string = PaymentSavedCredentials;

payments.paymentForm#3f56aea3 flags:# can_save_credentials:flags.2?true password_missing:flags.3?true bot_id:in

---functions---

payments.getPaymentForm#99f09745 msg_id:int = payments.PaymentForm;
```

If the user clicks on the keyboardButtonBuy button, the client proceeds to call payments.getPaymentForm with the message ID of the invoice preview message to get the payment form.

The returned form will contain fields that should be passed to the payment provider along with the full invoice.
The payment form also contains info about previously saved payment credentials and order information (name, phone number, email, shipping address & so on).

The full invoice contains info about the information required for the order, the price and the currency, and whether this is a `test` order.

2.3 Verifying information

```
invoice#c30aa358 flags:# test:flags.0?true name_requested:flags.1?true phone_requested:flags.2?true email_reque

postAddress#1e8caaeb street_line1:string street_line2:string city:string state:string country_iso2:string post_

paymentRequestedInfo#909c3f94 flags:# name:flags.0?string phone:flags.1?string email:flags.2?string shipping_ad

payments.validatedRequestedInfo#d1451883 flags:# id:flags.0?string shipping_options:flags.1?Vector<ShippingOpti

---functions---

payments.validateRequestedInfo#770a8e74 flags:# save:flags.0?true msg_id:int info:PaymentRequestedInfo = paymen
```

If any data at all is requested by the **invoice** (`name_requested`, `phone_requested`, `email_requested`,

usual, `msg_id` is the ID of the invoice message).
The user can choose to save order information for future use by setting the `save` flag.
Data can be autofilled as described in autofill.

If no errors are found in the submitted info, the response of the method will contain an `id` flag, to be used later to complete the payment.

If the `flexible` flag of the invoice is set, calling the payments.validateRequestedInfo method will send a shipping query update to the bot, to which the bot will reply with the available shipping options for the specified address as described here ».
The return value in this case will also contain a `shipping_options` field with the available shipping options.

If any errors are found in the submmitted data, a service notification will be sent to the user, with a description of the error from the bot.

2.3.1 Autofill

```
payments.savedInfo#fb8fe43c flags:# has_saved_credentials:flags.1?true saved_info:flags.0?PaymentRequestedInfo

---functions---

payments.getSavedInfo#227d824b = payments.SavedInfo;
payments.clearSavedInfo#d83d70c1 flags:# credentials:flags.0?true info:flags.1?true = Bool;
```

The requested fields can be autofilled with the info provided in the `saved_info` field of the payment form, or with the info fetched manually using payments.getSavedInfo.

Saved order information can also be cleared using payments.clearSavedInfo.

2.4 Select delivery option

```
labeledPrice#cb296bf8 label:string amount:long = LabeledPrice;

shippingOption#b6213cdf id:string title:string prices:Vector<LabeledPrice> = ShippingOption;

updateBotShippingQuery#e0cdc940 query_id:long user_id:int payload:bytes shipping_address:PostAddress = Update;

---functions---

messages.setBotShippingResults#e5f672fa flags:# query_id:long error:flags.0?string shipping_options:flags.1?Vec
```

If a shipping address was requested and the bot included the parameter `flexible`, when the user validates order information the Telegram API will send an updateBotShippingQuery to the bot.
The bot must respond using messages.setBotShippingResults either with a list of possible delivery options and the relevant delivery prices, or with an error (for example, if delivery to the specified address is not possible).

The returned shipping options or the shipping error will be returned to the user while validating order information.

3. Payment

```
inputPaymentCredentialsSaved#c10eb2cf id:string tmp_password:bytes = InputPaymentCredentials;
inputPaymentCredentials#3417d728 flags:# save:flags.0?true data:DataJSON = InputPaymentCredentials;
inputPaymentCredentialsApplePay#aa1c39f payment_data:DataJSON = InputPaymentCredentials;
inputPaymentCredentialsAndroidPay#ca05d50e payment_token:DataJSON google_transaction_id:string = InputPaymentCr


payments.paymentForm#3f56aea3 flags:# can_save_credentials:flags.2?true password_missing:flags.3?true bot_id:in
```

3.1 Web payment

```
inputPaymentCredentials#3417d728 flags:# save:flags.0?true data:DataJSON = InputPaymentCredentials;

payments.paymentForm#3f56aea3 flags:# can_save_credentials:flags.2?true password_missing:flags.3?true bot_id:in
```

Typically, payment takes place by opening the `url` in the specified payment form, which leads to a payment form on the website of the payment gateway.
Once the user finishes entering their payment credentials, a `payment_form_submit` web event is generated by the payment gateway, containing `data` and `title` JSON fields.

The `title` is used by the client app to represent the payment credentials (typically a censored version of credit card information).
The `data` is used to generate an inputPaymentCredentials constructor.
Eventually, you can set the `save` flag to save the credit card info for future use, only if 2FA is enabled.

Telegram **does not** have access to your card information. Credit card details will be handled only by the payment system.

3.2 Native payment

```
inputPaymentCredentials#3417d728 flags:# save:flags.0?true data:DataJSON = InputPaymentCredentials;

payments.paymentForm#3f56aea3 flags:# can_save_credentials:flags.2?true password_missing:flags.3?true bot_id:in
```

Most telegram apps support working natively with the native APIs of some payment providers, without opening the website of the payment and receiving a JS event.

This is done using the JSON `native_params` parameters field of the payments.paymentForm constructor, which contains an object, which can contain one or more of the following fields:

- `publishable_key` : Stripe API publishable key
- `apple_pay_merchant_id` : Apple Pay merchant ID
- `android_pay_public_key` : Android Pay public key
- `android_pay_bgcolor` : Android Pay form background color
- `android_pay_inverse` : Whether to use the dark theme in the Android Pay form
- `need_country` : True, if the user country must be provided,
- `need_zip` : True, if the user ZIP/postal code must be provided,
- `need_cardholder_name` : True, if the cardholder name must be provided

The payment gateway to use is decided based on the value of the `native_provider` field.

3.2.1 Stripe

```
inputPaymentCredentials#3417d728 flags:# save:flags.0?true data:DataJSON = InputPaymentCredentials;

payments.paymentForm#3f56aea3 flags:# can_save_credentials:flags.2?true password_missing:flags.3?true bot_id:in
```

If the `native_provider` field is set and equal to `stripe`, the client can make use of the native Stripe token APIs with the `publishable_key` from the `native_params` to add a payment method to Stripe, and then use the token `type` and `id` to generate a JSON object:

```
{"type":"token.type", "id":"token.id"}"
```

The generated JSON object can then be passed to the `data` field of the inputPaymentCredentials.
Eventually, you can set the `save` flag to save the credit card info for future use, only if 2FA is enabled.

Telegram **does not** have access to your card information. Credit card details will be handled only by the payment system.

Example implementation: Telegram for Android.

3.3 Apple pay

```
inputPaymentCredentialsApplePay#aa1c39f payment_data:DataJSON = InputPaymentCredentials;
```

On iOS devices, Apple Pay can be used to generate payment data, which is then sent using the inputPaymentCredentialsApplePay constructor.

Example implementation: Telegram for iOS.

3.4 Android pay

```
inputPaymentCredentialsAndroidPay#ca05d50e payment_token:DataJSON google_transaction_id:string = InputPaymentCr
```

On Android devices, Google Pay can be used to generate payment data, which is then sent using the inputPaymentCredentialsAndroidPay constructor.

Example implementation: Telegram for Android.

3.5 Using saved payment credentials

```
inputPaymentCredentialsSaved#c10eb2cf id:string tmp_password:bytes = InputPaymentCredentials;

paymentSavedCredentialsCard#cdc27a1f id:string title:string = PaymentSavedCredentials;

payments.paymentForm#3f56aea3 flags:# can_save_credentials:flags.2?true password_missing:flags.3?true bot_id:in

account.tmpPassword#db64fd34 tmp_password:bytes valid_until:int = account.TmpPassword;

---functions---

account.getTmpPassword#449e0b51 password:InputCheckPasswordSRP period:int = account.TmpPassword;
```

To reuse saved payment methods, the `saved_credentials` field of the payment form is used.
The `title` of the paymentSavedCredentialsCard can be used to preview a censored version of credit card info.
The `id` field is provided by the payment provider directly to the Telegram servers when saving the payment method, and identifies the payment method.
Full credit card info **is not** saved on Telegram Servers, and cannot be fetched by the user.

In order to **use** the saved payment method, 2FA must be enabled: the user must verify their identity by entering their 2FA password, which is then used as described in the SRP docs to generate SRP parameters which must be passed to account.getTmpPassword.

The generated temporary password can then be used to make payments using the saved credentials using the inputPaymentCredentialsSaved constructor.
The `id` field is the paymentSavedCredentialsCard `id`. The `tmp_password` is the temporary payment password generated by the server, if the user provided a correct 2FA password.

Example implementation: Telegram for Android.

4. Pre-Checkout

```
inputPaymentCredentialsSaved#c10eb2cf id:string tmp_password:bytes = InputPaymentCredentials;
inputPaymentCredentials#3417d728 flags:# save:flags.0?true data:DataJSON = InputPaymentCredentials;
inputPaymentCredentialsApplePay#aa1c39f payment_data:DataJSON = InputPaymentCredentials;
inputPaymentCredentialsAndroidPay#ca05d50e payment_token:DataJSON google_transaction_id:string = InputPaymentCr

payments.paymentResult#4e5f810d updates:Updates = payments.PaymentResult;
payments.paymentVerificationNeeded#d8411139 url:string = payments.PaymentResult;

---functions---

payments.sendPaymentForm#2b8879b3 flags:# msg_id:int requested_info_id:flags.0?string shipping_option_id:flags.
```

After verifying order information, the final step for the client is to call payments.sendPaymentForm, with the following parameters:

- The `msg_id` is set to the ID of the invoice message
- `requested_info_id` is set to the `id` of the verified order information, if it was requested
- `shipping_option_id` is set to the selected delivery option, if shipping was requested.
- `credentials` are the payment credentials generated by the payment provider, required to complete the order.

Payment method info can also be saved to the Telegram Servers and reused, by setting the `save` flag of inputPaymentCredentials when sending the form.
This is only possible on accounts with 2FA enabled.

The bot then replies to the received precheckout query, finally the user proceeds to checkout.

Please note that if the result of the method is a payments.paymentVerificationNeeded, before proceeding to checkout the payment provider requires the user to verify his identity by opening the provided `url` and following instructions. Once the user finishes working with the webpage, the client can proceed to checkout.

Eventual errors are returned in the form of RPC errors, with the description of the error by the bot contained in service updates.

4.1 Receiving pre-checkout query

```
paymentRequestedInfo#909c3f94 flags:# name:flags.0?string phone:flags.1?string email:flags.2?string shipping_ad

updateBotPrecheckoutQuery#5d2f3aa9 flags:# query_id:long user_id:int payload:bytes info:flags.0?PaymentRequeste

---functions---

messages.setBotPrecheckoutResults#9c2dd95 flags:# success:flags.1?true query_id:long error:flags.0?string = Boo
```

The user enters their payment information as described above and presses the final pay button.
At this moment the Telegram API sends an updateBotPrecheckoutQuery constructor that contains all the available information about the order to the bot.
The bot must reply using messages.setBotPrecheckoutResults **within 10 seconds** after receiving this update or the transaction is canceled.

The bot may return an error if it can't process the order for any reason. We highly recommend specifying a reason for failure to complete the order in human readable form (e.g. "Sorry, we're all out of rubber ducks! Would you be interested in a steel bear instead?"). Telegram will display this reason to the user.

5. Checkout

```
keyboardButtonBuy#afd93fbb text:string = KeyboardButton;

keyboardButtonRow#77608b83 buttons:Vector<KeyboardButton> = KeyboardButtonRow;
replyInlineMarkup#48a30254 rows:Vector<KeyboardButtonRow> = ReplyMarkup;

messageMediaInvoice#84551347 flags:# shipping_address_requested:flags.1?true test:flags.3?true title:string des

message#44f9b43d flags:# out:flags.1?true mentioned:flags.4?true media_unread:flags.5?true silent:flags.13?true

updateNewMessage#1f2b0afd message:Message pts:int pts_count:int = Update;

payments.paymentReceipt#500911e1 flags:# date:int bot_id:int invoice:Invoice provider_id:int info:flags.0?Payme

---functions---

payments.getPaymentReceipt#a092a980 msg_id:int = payments.PaymentReceipt;
```

In case the bot confirms the order, Telegram requests the payment provider to complete the transaction. If the

payment information was entered correctly and the payment goes through, the Telegram API will modify the invoice message and send a service message about it. Only after this should the bot should proceed with delivering the goods or services purchased by the user.

If all is OK, the user receives a payments.paymentResult in reply to the payments.sendPaymentForm query, containing info about the updated invoice message in the form of an updateEditMessage.

The invoice message will be updated as follows: the attached messageMediaInvoice will now have a `receipt_msg_id` field.
Clients should treat invoice messages with a `receipt_msg_id` field as receipt messages, **locally** modifying the label of the keyboardButtonBuy button to a localized version of the word `Receipt`.
From this point, clicking on the `Receipt` button should trigger a call to payments.getPaymentReceipt, providing the `receipt_msg_id` to the `msg_id` field, which will return info about the transaction.

The payment will also generate one service message of type messageActionPaymentSent or messageActionPaymentSentMe, replying to the invoice.
For bots, the service message will be of type messageActionPaymentSentMe, for users it will be a messageActionPaymentSent.

```
messageActionPaymentSentMe#8f31b327 flags:# currency:string total_amount:long payload:bytes info:flags.0?Paymen
messageActionPaymentSent#40699cd0 currency:string total_amount:long = MessageAction;
```

Styled text with message entities

Telegram supports styled text using message entities.

A client that wants to send styled messages would simply have to integrate a Markdown/HTML parser, and generate an array of message entities by iterating through the parsed tags.

Special care must be taken to consider the UTF-8 length of strings when generating message entities, see example implementations: tdlib, MadelineProto.

Nested entities are supported, for example the following HTML/Markdown aliases for message entities can be used:

- **messageEntityBold** => `<b>bold</b>`, `<strong>bold</strong>`, `**bold**`
- *messageEntityItalic* => `<i>italic</i>`, `<em>italic</em>` `*italic*`
- `messageEntityCode` => `<code>code</code>`, `` `code` ``
- ~~messageEntityStrike~~ => `<s>strike</s>`, `<strike>strike</strike>`, `<del>strike</del>`, `~~strike~~`
- messageEntityUnderline => `<u>underline</u>`
- `messageEntityPre` => `<pre language="c++">code</pre>`,

```c++
code
```

Message drafts

Message drafts in Telegram allow syncing the text typed into message fields between devices.

Drafts

Drafts are represented by the DraftMessage constructors.
The parameters of the peer-specific draft should be used as defaults when composing a message to be sent to a certain peer (in the case of media, the same draft should still be used as base, the message will become the caption).
If the user exits the app before sending the message, the message should be saved as a draft:

Saving drafts

Drafts can be saved using the messages.saveDraft method.

Downloading drafts

New drafts are automatically sent to all devices via updateDraftMessage updates.

Dialog objects fetched via the API also contain the draft associated with the dialog.

Clearing drafts

Drafts can be cleared by setting the `clear_draft` flag when sending messages or media using messages.sendMessage, messages.sendMedia, messages.sendMultiMedia, messages.sendInlineBotResult and similar or manually by passing empty values to messages.saveDraft.

Top peer rating

If enabled, the rating of top peers indicates the relevance of a frequently used peer in a certain category (frequently messaged users, frequently used bots, inline bots, frequently visited channels and so on).

The rate delta is computed by taking the time delta between the last time the user used a certain peer and the last time the rating for that peer was received and dividing it by the exponential decay from config.

Example:
Client-side, every time a user opens chat `123456789` the following operation must be done on the cached top peer info.

- `dateOpened` indicates when was the peer used
- `normalizeRate` is an arbitrary time in the recent past.
  When ratings are received from the server it is the time when they were received.

```
topPeer.rating += e^((dateOpened - normalizeRate) / config.rating_e_decay)
```

File references

File references are strings of bytes, that can be encountered in the `file_reference` fields of document and photo objects.

They must be cached by the client, along with the **origin context** where the document/photo object was found, in order to be refetched when the file reference expires.

Example implementation of a reference database: MadelineProto, android, telegram desktop, tdlib.

Another example:

Assume you receive a message from your friend: that message contains a messageMediaPhoto with a photo.

Your client has to cache not only the `file_reference` field of the photo, but also the context in which the file reference was seen (in this case, a message coming from a specific user).

The context info is in this case, an origin context of type message, containing the message ID and the peer ID of the chat/channel/user where the message was seen.

The context info has to be associated with the file reference: when downloading a file using upload.getFile, a `FILE_REFERENCE_EXPIRED` error (or another error starting with `FILE_REFERENCE_`) may be returned.
If this happens, the context info must be used to refetch the object that contained the file reference: in this example, the peer info and the message ID have to be used with channels.getMessages or messages.getMessages to refetch the message, recache the file reference and use it in a new file download request.

More than one origin context can be associated to one file reference, for greater resilience (in the case of a message that was deleted in one chat but was also forwarded in another chat, the file reference can be refetched from the second chat, instead).

Origin contexts for objects returned by method calls with certain parameters can be considered, too (for example, in the case of favorited sticker sets returned by messages.getFavedStickers).

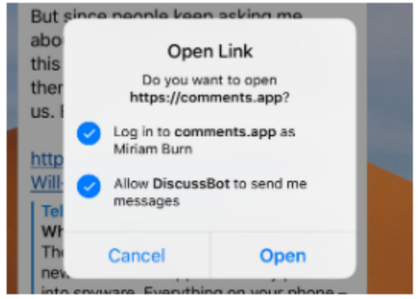
Seamless Telegram Login

Bots may ask users to login to a certain website via Telegram when clicking on certain URL buttons in inline keyboards.

When the user clicks on keyboardButtonUrlAuth, messages.requestUrlAuth should be called, providing the `button_id` of the button and the ID and peer of the container message.
The returned urlAuthResultRequest object will contain more details about the authorization request:

- The `domain` parameter will contain the domain name of the website on which the user will log in (example: *comments.app*).
- The `bot` parameter will contain info about the bot which will be used for user authorization (example: *DiscussBot*).
- The `request_write_access` will be set if the bot would like to send messages to the user.

The info should be shown in a prompt:



If the user agrees to login to the URL, messages.acceptUrlAuth should be called (eventually setting the `write_allowed` if the permission was requested and the user consented).
The result will be a urlAuthResultAccepted with the final URL to open, which will include a query string with the requested info and a hash that must be verified upon receival by the service.

urlAuthResultDefault could also be returned, instead, in which case the `url` of the keyboardButtonUrlAuth must be opened, instead.
The same must be done if the user opens the link while refusing the authorization request.

Web events

When interacting with HTML5 games and the websites of payment gateways, Telegram apps should expose APIs to allow receiving data and events from the websites.

Event APIs

Games and payment gateways can generate events that are meant to be received by the Telegram apps.
Typically events are generated by using the `postEvent` method of the GamingCommunication library.
The `postEvent` function will try sending the event to the Telegram app in a number of different ways.

WebviewProxy

In mobile apps, the event receiver API should be typically exposed as a `window.TelegramWebviewProxy` object with a `postEvent` method.

```
window.TelegramWebviewProxy.postEvent(eventType, eventData)
```

window.external

Alternatively, a `window.external.notify` method can be exposed, accepting a string JSON payload with the event type and payload:

```
window.external.notify(JSON.stringify({eventType: eventType, eventData: eventData}));
```

postMessage API

Finally, web MTProto clients that need to open a game or process a payment in an iframe can use the postMessage API to receive events from iframes.
The GamingCommunication library by defaultwill use `'*'` as `targetOrigin`, sending messages to parent pages regardless of the origin of the embedder.

```
window.parent.postMessage(JSON.stringify({eventType: eventType, eventData: eventData}), targetOrigin);
```

Event types

`eventType` is a simple string indicating the event type, and `eventData` is a payload with an object that will be parsed by the Telegram app.

eventType	eventData	Description
`payment_form_submit`	JSON object with `data` and `title` fields	`title` is the censored credit card title. `data` is a service-specific JSON payload with information about the payment credentials provided by the user to the payment system. **Neither Telegram, nor bots will have access to your credit card information.** Credit card details will be handled only by the payment system.
`share_score`	null	Will be called by games when the user explicitly clicks on the **share score** button to share the game, along with his score. Typically done by using messages.forwardMessages on the game message with the `with_my_score` flag.
`share_game`	null	Will be called by games when the user explicitly clicks on the **share game** button to share the game, without sharing his score. Typically done by using messages.forwardMessages on the game message without the `with_my_score` flag, or by sharing the game's short URL.
`game_over`	null	Can be called by games when the user loses a game
`game_loaded`	null	Can be called by games once the game fully loads
`resize_frame`	JSON object with `height` field	Called by supported pages inside of IV iframe embeds, indicates the new size of the embed frame.

MTProto Mobile Protocol

> Please feel free to check out our FAQ for the Technically Inclined.
> Client developers are required to comply with the Security Guidelines.

Related articles

Mobile Protocol: Detailed Description
Creating an Authorization Key
Creating an Authorization Key: Example
Mobile Protocol: Service Messages
Mobile Protocol: Service Messages about Messages
Binary Data Serialization
TL Language

MTProto TL-schema

End-to-end encryption, Secret Chats
End-to-end TL-schema
Security Guidelines for Client Software Developers

This page deals with the basic layer of MTProto encryption used for Cloud chats (server-client encryption). See also:

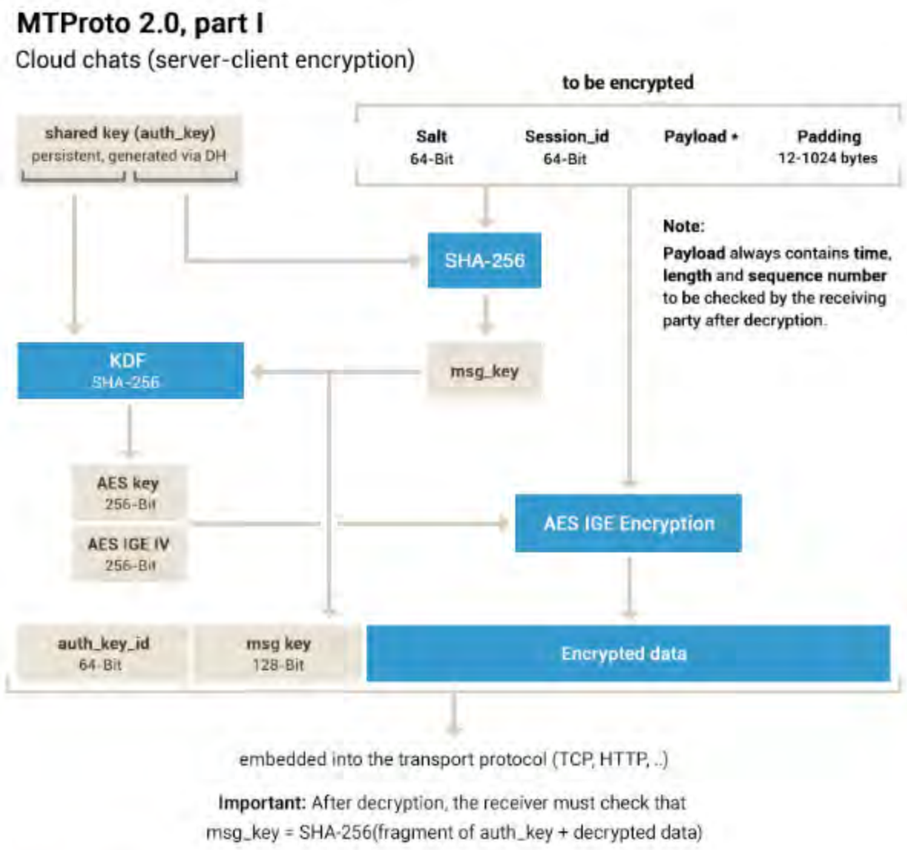
- Secret Chats, end-to-end-encryption
- End-to-end encrypted Voice Calls

General Description

The protocol is designed for access to a server API from applications running on mobile devices. It must be emphasized that a web browser is not such an application.

The protocol is subdivided into three virtually independent components:

- High-level component (API query language): defines the method whereby API queries and responses are converted to binary *messages*.
- Cryptographic (authorization) layer: defines the method by which messages are encrypted prior to being transmitted through the transport protocol.
- Transport component: defines the method for the client and the server to transmit messages over some other existing network protocol (such as HTTP, HTTPS, WS (plain websockets), WSS (websockets over HTTPS), TCP, UDP).



> As of version 4.6, major Telegram clients are using **MTProto 2.0**, described in this article.
> MTProto v1.0 (described here for reference) is deprecated and is currently being phased out.

Brief Component Summary

High-Level Component (RPC Query Language/API)

From the standpoint of the high-level component, the client and the server exchange *messages* inside a *session*. The session is attached to the client device (the application, to be more exact) rather than a specific websocket/http/https/tcp connection. In addition, each session is attached to a *user key ID* by which authorization is actually accomplished.

Several connections to a server may be open; messages may be sent in either direction through any of the connections (a response to a query is not necessarily returned through the same connection that carried the original query, although most often, that is the case; however, in no case can a message be returned through a connection belonging to a different session). When the UDP protocol is used, a response might be returned by a different IP address than the one to which the query had been sent.

There are several types of messages:

- RPC calls (client to server): calls to API methods
- RPC responses (server to client): results of RPC calls
- Message received acknowledgment (or rather, notification of status of a set of messages)
- Message status query
- *Multipart message* or *container* (a container that holds several messages; needed to send several RPC calls at once over an HTTP connection, for example; also, a container may support gzip).

From the standpoint of lower level protocols, a message is a binary data stream aligned along a 4 or 16-byte boundary. The first several fields in the message are fixed and are used by the cryptographic/authorization system.

Each message, either individual or inside a container, consists of a *message identifier* (64 bits, see below), a *message sequence number within a session* (32 bits), the *length* (of the message body in bytes; 32 bits), and a *body* (any size which is a multiple of 4 bytes). In addition, when a container or a single message is sent, an *internal header* is added at the top (see below), then the entire message is encrypted, and an *external header* is placed at the top of the message (a 64-bit *key identifier* and a 128-bit *message key*).

A *message body* normally consists of a 32-bit *message type* followed by type-dependent *parameters*. In particular, each RPC function has a corresponding message type. For more detail, see Binary Data Serialization, Mobile Protocol: Service Messages.

All numbers are written as little endian. However, very large numbers (2048-bit) used in RSA and DH are written in the big endian format because that's how the OpenSSL library does it.

Authorization and Encryption

Prior to a message (or a multipart message) being transmitted over a network using a transport protocol, it is encrypted in a certain way, and an *external header* is added at the top of the message which is: a 64-bit *key identifier* (that uniquely identifies an *authorization key* for the server as well as the *user*) and a 128-bit *message key*. A user key together with the message key defines an actual 256-bit key which is what encrypts the message using AES-256 encryption. Note that the initial part of the message to be encrypted contains variable data (session, message ID, sequence number, server salt) that obviously influences the message key (and thus the AES key and iv). The message key is defined as the 128 middle bits of the SHA256 of the message body (including session, message ID, etc.), including the padding bytes, prepended by 32 bytes taken from the authorization key. Multipart messages are encrypted as a single message.

> For a technical specification, see Mobile Protocol: Detailed Description

The first thing a client application must do is create an authorization key which is normally generated when it is first run and almost never changes.

The protocol's principal drawback is that an intruder passively intercepting messages and then somehow appropriating the authorization key (for example, by stealing a device) will be able to decrypt all the intercepted messages *post factum*. This probably is not too much of a problem (by stealing a device, one could also gain access to all the information cached on the device without decrypting anything); however, the following steps could be taken to overcome this weakness:

- *Session keys* generated using the Diffie-Hellman protocol and used in conjunction with the authorization and the message keys to select AES parameters. To create these, the first thing a client must do after creating a new session is send a special RPC query to the server ("generate session key") to which the server will respond, whereupon all subsequent messages within the session are encrypted using the session key as well.
- Protecting the key stored on the client device with a (text) password; this password is never stored in memory and is entered by a user when starting the application or more frequently (depending on application settings).
- Data stored (cached) on the user device can also be protected by encryption using an authorization key which, in turn, is to be password-protected. Then, a password will be required to gain access even to that data.

Time Synchronization

If client time diverges widely from server time, a server may start ignoring client messages, or vice versa, because of an invalid message identifier (which is closely related to creation time). Under these circumstances, the server will send the client a special message containing the correct time and a certain 128-bit salt (either explicitly provided by the client in a special RPC synchronization request or equal to the key of the latest message received from the client during the current session). This message could be the first one in a container that includes other messages (if the time discrepancy is significant but does not as yet result in the client's messages being ignored).

Having received such a message or a container holding it, the client first performs a time synchronization (in effect, simply storing the difference between the server's time and its own to be able to compute the "correct" time in the future) and then verifies that the message identifiers for correctness.

Where a correction has been neglected, the client will have to generate a new session to assure the monotonicity of message identifiers.

MTProto transport

Before being sent using the selected transport protocol, the payload has to be wrapped in a secondary protocol header, defined by the appropriate MTProto transport protocol.

- Abridged
- Intermediate
- Padded intermediate
- Full

The server recognizes these different protocols (and distinguishes them from HTTP, too) by the header.
Additionally, the following transport features can be used:

- Quick ack
- Transport errors
- Transport obfuscation

Example implementations for these protocols can be seen in tdlib and MadelineProto.

Transport

Enables the delivery of encrypted containers together with the external header (hereinafter, *Payload*) from client to server and back.
Multiple transport protocols are defined:

- TCP
- Websocket
- Websocket over HTTPS
- HTTP
- HTTPS
- UDP

(We shall examine only the first five types.)

Recap

To recap, using the ISO/OSI stack as comparison:

- Layer 7 (Application): High-level RPC API
- Layer 6 (Presentation): Type Language
- Layer 5 (Session): MTProto session
- Layer 4 (Transport):
  - 4.3: MTProto transport protocol
  - 4.2: MTProto obfuscation (optional)
  - 4.1: Transport protocol
- Layer 3 (Network): IP
- Layer 2 (Data link): MAC/LLC
- Layer 1 (Physical): IEEE 802.3, IEEE 802.11, etc...

Jobs

Telegram stands for **freedom** and **perfection**. It is built by a tiny team of world-renowned developers selected in multi-level contests. If you want to join the **Telegram team**, check out the options below.

C/C++ Software Engineer

Responsibilities: work on Telegram's custom-built low-level data storage engines distributed across several data-centers.

Preferred qualifications: experience working on scalable C/C++ projects and/or networking protocols, experience working with Linux, top results in programming contests.

How to apply: contact @jobs_bot and fill in the required info. If you are selected, the bot will share a test task in the coming weeks.

Voice / Video Calls Developer

Responsibilities: constantly measure and improve the quality of Telegram voice calls. Develop automated tests to benchmark Telegram voice calls against the competition.

Preferred qualifications: deep knowledge of VoIP protocols and codecs, experience working with Android Java and/or iOS Swift.

How to apply: contact @jobs_bot and fill in the required info. In January, the bot will inform you about an upcoming coding contest, the winner of which will receive a prize (~50,000 USD) and an opportunity to join our team.

Android Software Engineer (Camera, Video)

Responsibilities: develop client-side camera and video compression software for Telegram android apps.

Preferred qualifications: experience developing in Java for Android, proactive perseverance in maximizing speed and quality on the widest range of Android devices.

How to apply: contact @jobs_bot and fill in the required info. The bot will inform you about an upcoming competition, the winner of which will receive a prize (~40,000 USD) and an opportunity to join our team.

Motion Graphic Artist / Animator

Responsibilities: develop UI animations for Telegram mobile applications, design animated characters.

Preferred qualifications: vector illustration skills, motion design and animated character design skills.

How to apply: contact @jobs_bot and fill in the required info. In January 2019, the bot will inform you about an upcoming artists' competition, the winner of which will receive a prize (~20,000 USD) and may be invited to join our team.

Site Reliability Engineer

Responsibilities: automate routine tasks, proactively identify and solve potential reliability issues, increase fault-tolerance of Telegram's multi-datacenter infrastructure.

Preferred qualifications: experience administering *nix like systems, experience developing in C, Python or Perl; knowledge of bash, network protocols, and network equipment of major manufacturers.

How to apply: contact @jobs_bot and fill in the required info. If you are selected, the bot will share a test task in the coming weeks.

Javascript Developer

Responsibilities: help build Telegram for Web.

Preferred qualifications: experience developing fast and efficient JS-code.

How to apply: contact @jobs_bot and fill in the required info. The bot will inform you about a competition in the coming months.

Writer

Responsibilities: help write Telegram blog posts, tweets, FAQs and all sorts of other Telegram-related texts with elegance, humor and style.

Preferred qualifications: strong analytical and editing skills, experience in producing high-quality texts under ambitious deadlines.

How to apply: contact @jobs_bot and fill in the required info. The bot will share a test task in the coming weeks.

Junior Accountant

Responsibilities: maintain accounting books and records, ensure documentation of financial transactions, manage payrolls, prepare financial statements and reports (balance sheet, profit and loss statement, cash flow and analytics).

Preferred qualifications: high GMAT score, native-level fluency in English, appreciation and working knowledge of accounting principles under IFRS, experience with QuickBooks or Xero software, strong computer literacy with proficient use of Microsoft Excel.

How to apply: contact @jobs_bot and fill in the required info. If you are selected, we will get in touch with you in the coming weeks.

Junior Legal Counsel

Responsibilities: draft and assess all types of legal documents and legal correspondence, ensure compliance with applicable laws, proactively conduct legal research, keep up with changes in laws and regulations.

Preferred qualifications: strong legal training in the UK/US, in-depth knowledge of administrative law and procedures, analytical skills, exposure to legal frameworks in multiple jurisdictions, excellent communication skills with the ability to clearly and concisely express ideas both verbally and in writing.

How to apply: contact @jobs_bot and fill in the required info. If you are selected, the bot will share a test task in the coming weeks.

Assistant to the CEO

Responsibilities: manage and maintain calendar of the CEO, booking and arranging travel, transport and accommodation, lifestyle management.

Preferred qualifications: high IQ score, native-level fluency in English, excellent verbal and written communications skills, accuracy and attention to detail, basic financial skills, discretion, persistence.

How to apply: contact @jobs_bot and fill in the required info. If you are selected, we will get in touch with you in the coming weeks.

Design Competition

There are currently two design competitions held by Telegram.

UI Design: https://t.me/designers/17
($15,000 prize)
Animation Design: https://t.me/designers/18
($10,000 prize)

Dates: 10 March 2019 – 24 March 2019

Instant View Template Competition

Goal: create sets of rules to parse website content to present it in a light and readable way to Telegram users.

Preferred qualifications: experience working with HTML/CSS layouts.

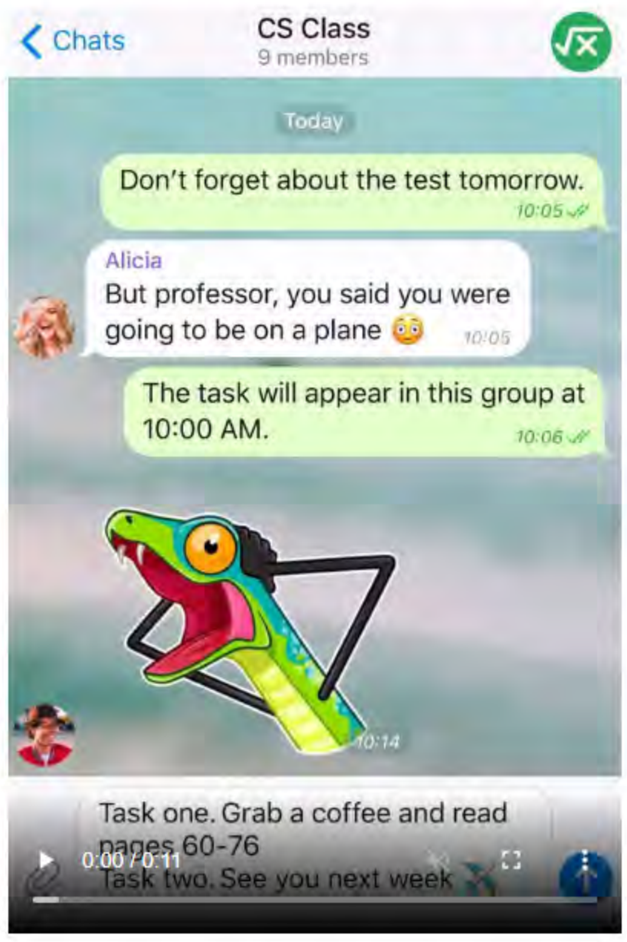
Dates: 1 Feb 2019 – 31 March 2019

More about Instant View

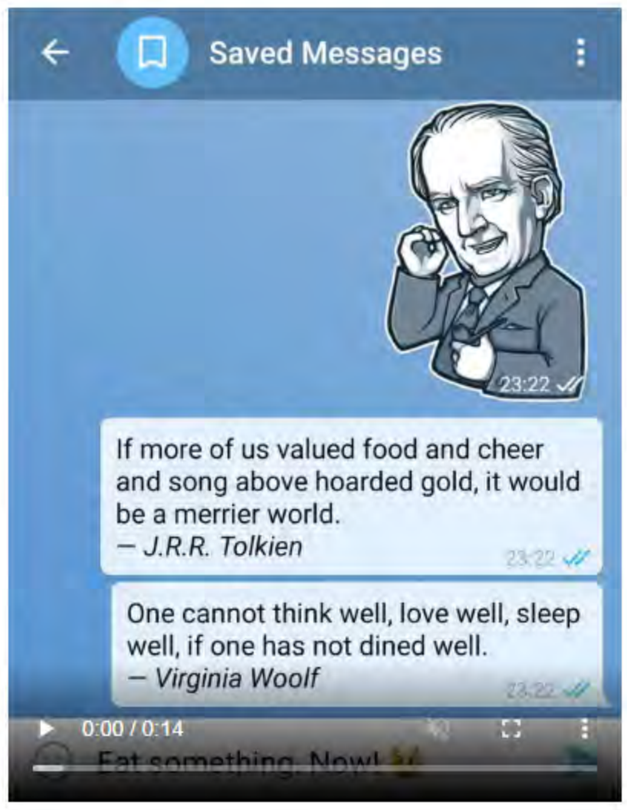
Scheduled Messages, Reminders, Custom Cloud Themes and More Privacy

In our previous update, we added a handy menu when you hold the send button. Today we're populating this menu with a new option that helps you plan ahead.

Hold the *'Send'* button in any chat and select **'Schedule Message'** to automatically send things at a specified time in the future (DeLorean not included).



Scheduling also works in your *'Saved Messages'* chat, turning your planned posts into **reminders**. Be your own time-travelling secretary - whether it's about a dentist appointment next week, or waking up in time for pizza delivery.

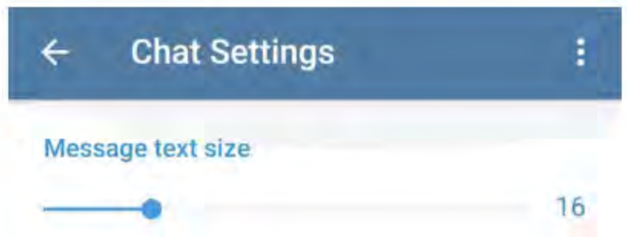


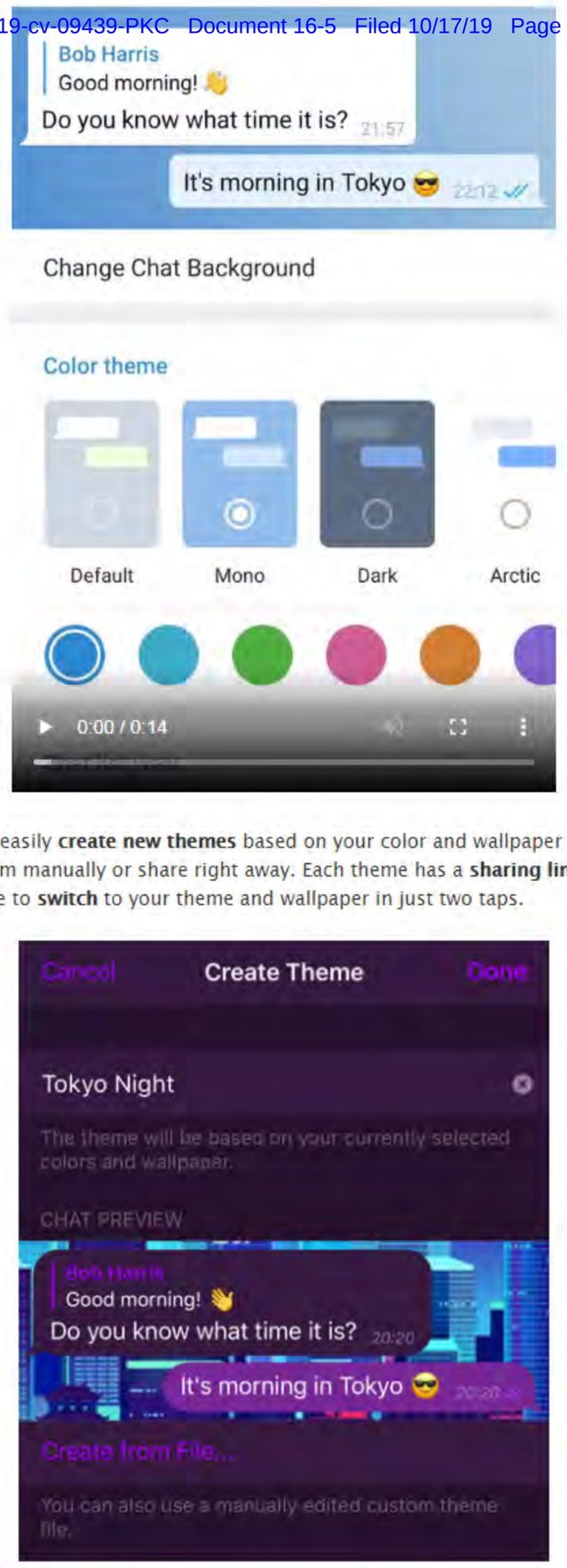
Whenever a scheduled message or reminder is sent, you get a special **notification** marked with a 📅, so you don't get caught off-guard by messages you planned in the past.

Custom Cloud Themes

You could first **customize Telegram's appearance** way back in 2017. Today's update makes this easier than ever across all platforms, including our native desktop apps.

Choose a pre-set **accent color** or pick something unique from the **color wheel**, and the app will adjust all elements accordingly. See how it looks in your favorite color, blue - no, yellow!





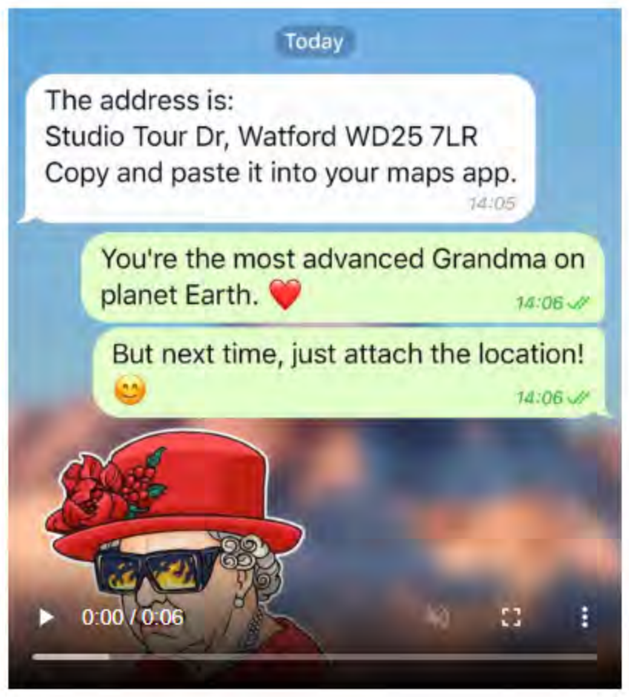
You can now easily **create new themes** based on your color and wallpaper choices, then fine-tune them manually or share right away. Each theme has a **sharing link** which allows anyone to **switch** to your theme and wallpaper in just two taps.

When you change your theme, it gets **updated** for all its users. This way you can update your app's looks to evolve as Telegram updates – or to change with the seasons.

To try out cloud themes, take a look at Sky Blue or Desert. These will work for both iOS and Android, and we solemnly swear to update them to acid green on orange when you least expect it.

Redesigned Message Options

On iOS, we have condensed message options into a **single menu** for forwarding, replying and more. You can now also **select any portion of the message** to copy or share, instead of only the full text.



New Privacy Settings

Telegram groups can be **public** and may hold up to **200,000** members each. When launching them, we were thinking of campuses, conventions and spaces where you could properly brag about your cats. However, Telegram communities are also increasingly used by people to organize themselves in the face of oppression.

We believe that all people have a right to **express their opinions** and communicate **privately**. To further protect these rights, we're expanding Telegram's arsenal of **Privacy Settings** today.

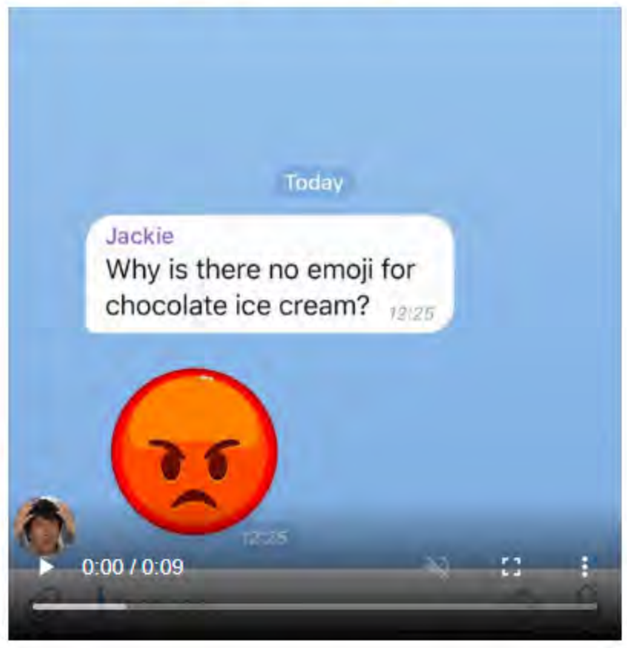
If you set *Who Can See My Phone Number* to *'Nobody'*, a new option will appear below, allowing you to control your visibility for those who **already have it**. Setting *Who Can Find Me By My Number* to *'My Contacts'* will ensure that random users who add your number as a contact are **unable to match** your profile to that number.

If, like the majority of our users, you rely on uploading phone contacts to identify friends and family members on the app, this setting won't get in your way.

More Animated Emoji

Regardless of whether you're fighting for freedom or queuing for coffee, you'll likely find a use for the new additions to our **animated emoji** family.

As before, send a single 😄, 😘, 😡, 💩, 🤫 or 🤔 to unleash their spirit in motion.



And that's all, folks! While waiting for our next update, check out this brief history of Telegram. Or, if you're not into reading, run a bath and enjoy these animated rubber duck stickers.

September 5, 2019
The Telegram Team

Forward Tweet

Celebrating 6 Years of Telegram

We launched Telegram **6 years** ago, on **August 14, 2013** – and immediately set to work on the next update. That's how it's been ever since.

After a little while we noticed that our work not only brings progress to Telegram – but also results in the **evolution of messaging** as a whole. It is heartwarming to see our solutions and design decisions adopted as de facto standards across the board.

To keep you entertained while we're working on this month's second update, we've compiled a **brief history of Telegram**. It ended up rather on the 'humongous' side of 'brief' – but it's not like the next update is tomorrow either. So read on.

Oh, and if you want to give us a gift, bring two of your best friends to Telegram today.

A (Not So) Brief History of Telegram

For some reason, we started writing this backwards – from the latest update to the primordial summer of **2013**. And then it was too late to change the order because the guy who compiled the post said he would go on a hunger strike if we tried.

So if you're a fan of chronology, please jump to the end and scroll up from there. (By the way, we decided we're going to keep this post up to date as new updates arrive.)

September 2019

Scheduled Messages, Reminders, Custom Cloud Themes and More Privacy

The end of summer brought **scheduled messages** and **reminders** to better keep track of important tasks and appointments. On the visual side, shareable **Custom Cloud Themes** came to all platforms, along with more animated emoji.

An important privacy option was added for those who need it most – **Who Can Find Me By My Phone Number**. This new setting can prevent random users (or government organizations) from being able to match a number from their device's contacts to your Telegram account.

On Telegram since Septebmer 2019:

- Scheduled Messages and Reminders
- Custom Cloud Themes
- Who Can Find Me By My Number
- Selective Text Copying on iOS

August 2019

Silent Messages, Slow Mode, Admin Titles and More

The August update brought a way to send **silent messages** – which result in a notification **without sound**. We also added **custom titles** and a **slow mode** for group admins. But you probably remember this, it's only been a week.

Following the success of animated stickers, this update also introduced **animated emoji** for ❤️, 👍, 😏, 😳 and 🥳. More followed.

On Telegram since August 2019:

- Silent messages
- Slow mode
- Admin titles
- Timestamp links for videos
- Improved video scrubbing
- Animated emoji
- New attachment menu on Android
- Accent colors for night mode on iOS
- Comments widget

July 2019

Animated Stickers Done Right

The July update introduced **animated stickers**. We managed to achieve a smooth **60 FPS** at just **20–30 KB** per sticker. Animated stickers consume less bandwith than static ones and less battery than GIFs, while commanding more attention than the two combined.

Here are some sample animated sets in case you missed them: Hot Cherry, Tidy Tie Tom, Resistance Dog, Fred the Pug, Melie the Cavy, Earl the Wolf, Egg Yolk

These stickers are powered with uranium and dreams and are guaranteed to keep going forever.

On Telegram since July 2019:

- Animated stickers
- Open platform for animated stickers

June 2019

Location-Based Chats, Adding Contacts Without Phone Numbers and More

The June update focused on making it easier to **add people to contacts**. A universal **Add** button was introduced to all new chats, making it possible to add users to contacts even if you don't know their number yet.

A new **Add People Nearby** section was created to quickly exchange numbers with people close to you. The new section also allows creating **location-based public chats**.

On Telegram since June 2019:

- Add contacts without phone numbers
- Add people nearby
- Location-based chats
- Transfer group ownership to other users
- Enhanced notification exceptions
- Siri shortcuts
- New theme picker on iOS
- Change your app icon on iOS

May 2019, Second Update

Focused Privacy, Discussion Groups, Seamless Web Bots and More

The second May update made Telegram Privacy Settings even more **flexible**. You can add whole **groups** to *'Never Share'* or *'Always Share'* exceptions. Settings will adjust automatically as people join and leave the groups.

On Telegram, your **phone number** was never visible to random people in groups, the default being *'My Contacts'*. This update offered even more control via *Settings > Privacy & Security > Phone Number*.

The update also introduced an easier way to integrate **web services** with Telegram, adding one-tap authorization via bots. Last but not least, it offered **two** new ways of **adding comments to channels** – via Comments.app and connected **discussion groups**. (We didn't start the flame war.)

On Telegram since May 31, 2019:

- Use groups as exceptions for Privacy Settings

- 'Who can see my phone number' setting
- Seamless authorization for web services via bots
- Discussion groups for Channels
- View public channels without logging in
- Add text URLs to messages on iOS (Android since June 2018)
- New theme switcher for Android

May 2019, First Update

Archived Chats, a New Design on Android and More

The first May update introduced **Archived Chats**. Unmuted chats pop out of the archive with the next incoming message – but muted chats stay archived forever (or until they generate a notification via a mention or reply). An **unlimited** number of chats can be pinned in the archive.

Android design got a **major overhaul**. Aside from the visual improvements, this update added a **three-lines-per-chat** view option for the Android chat list (similar to the iOS default), as well as **bulk actions** in the chats list to easily delete, archive or mute multiple chats at once.

On Telegram since May 9, 2019:

- Archived chats
- New design on Android
- Bulk actions in the chat list on Android
- Expanded view for chat list on Android
- Improved passcodes on iOS
- Message links in groups

March 2019

Taking Back Our Right to Privacy

Before the March update, you could delete your messages within 48 hours of sending them. As of this update, you can **delete any message** you've ever sent or received in **any** private chat, there's **no time limit**. This also applies to clearing entire chats. The deleted messages don't leave a mark in the chat.

This update also introduced a setting for **Anonymous Forwards**, allowing you to control when messages forwarded from your account link back to it. This way people you chat with have no verifiable proof you ever sent them anything.

On Telegram since March 2019:

- Delete any private message for both parties, anytime
- Anonymized forwarding
- Search for Settings and FAQ entries
- Search emoji by keyword
- VoiceOver and TalkBack support

February 2019

Autoplaying Videos, Automatic Downloads and Multiple Accounts

Following the February update, smaller videos **start playing** without sound when they reach your screen. Pressing the volume buttons on your device unmutes them.

This update also introduced **presets** for autodownload settings to switch between **Low**, **Medium**, **High** and **Custom** data consumption in a few taps. Default settings for data usage have become more generous but depend on the affordability of mobile data in each particular country.

As of this update, iOS users can also log in to Telegram with **multiple accounts** from one app, easily switch between them and get notifications. Support for multiple accounts was added to Android in December 2017.

On Telegram since February 2019

- Autoplaying videos
- Autodownload presets (Low, Medium, High)
- Adaptive autodownload settings
- Multiple account support on iOS (on Android since December '17)
- Logout alternatives screen

January 2019, Second Update

Chat Backgrounds 2.0: Make Your Own

The second update in January brought **chat wallpapers** to version **2.0**, adding support for **motion** and **blur** effects, **plain color** backgrounds and **patterns**, as well as an option to **share** backgrounds via links and find new ones in **search**.

used at launch can be found here.

On Telegram since January 31, 2019

- New backgrounds for chats, including photos
- Choose from photos, solid colors and patterns
- Motion and blur effects for backgrounds
- Share backgrounds via links
- Search for new backgrounds
- Instant View 2.0 contest finished

January 2019, First Update

Group Permissions, Undo Delete and More

The first January update brought **group permissions**, allowing admins to restrict all members from posting specific kinds of content or even from messaging altogether. The maximum size for groups was doubled to **200,000 members**.

The update also added an **'Undo'** option after **deleting chats** or clearing history. It's like a time machine, but with fewer paradoxes.

On Telegram since January 21, 2019

- Default permissions for groups
- Up to 200,000 members in each group
- 'Undo' deleting chats or clearing history
- Sort contacts by name or 'Last seen'
- Better thumbnails and new download animations
- Choose emoji sets on Telegram Desktop

2018

December 2018, Second Update

Polls: Bringing Choice to Communities

As of the second update from December '18, Telegram users can create **polls** in groups and channels, letting group members vote anonymously.

Poll bots have been available since the dawn of the Bot Platform.

On Telegram since December 22, 2018

- Polls for groups and channels

December 2018, First Update

Custom Languages, Instant View 2.0 and More

This first December update in 2018 revolutionized app interfaces, adding **custom languages** – whether that's Māori, Scottish Gaelic or Transliterated Klingon. Qapla'. Users can **share** translated **language packs** with dedicated sharing links.

Everyday be talk like a pirate day if ye've downloaded the pirate speak chest. Babel fish now optional.

On Telegram since December 10, 2018

- Create and share custom language packs
- Redesigned Android app
- Badge counter settings added on Android
- Crop profile photos on Android
- Zoom videos during playback on Android
- Instant View 2.0 supports tables, nested lists, RTL languages and more

October 2018

Introducing Telegram 5.0 for iOS

Telegram for iOS was **rebuilt from scratch** for this update – using the **Swift** programming language. The new client is slicker, faster and more battery-friendly. If the old Telegram was a race car, this one is a race car with jet engines and a well-caffeinated driver.

New **badge counter** settings were added on iOS, allowing users to exclude messages from muted chats or count **unread chats** instead of messages.

The native macOS client added **swipe gestures** to navigate interfaces. MacBook Pro owners got **touch bar support**.

On Telegram since October 2018

- New iOS app
- Expand in-app notifications
- Count unread chats instead of unread messages
- Skip to the beginning of the day's chat
- Swipe gestures for MacOS
- TouchBar support for Macbook Pro
- Local storage for Telegram Desktop

August 2018

Chat Export Tool, Better Notifications and More

This update brought an easy way to **download** copies of **your chats**, including **photos and other media** using Telegram Desktop. August '18 also saw the birth of **Notification Exceptions** – a way to customize notification settings for particular users.

On Telegram since August 2018

- Export chat history and account information
- Customized notifications for individual users
- Support for new document types in Telegram Passport
- Support for names in original language in Telegram Passport

July 2018

Introducing Telegram Passport

Telegram Passport was launched as a **unified authorization service** for ICOs, financial services and other platforms based on **real-life ID**. You no longer have to upload scans of your ID each time a different service requires it.

On Telegram since July 2018

- Telegram Passport
- E2E Encryption for ID storage

June 2018

Replace Media, Share vCards, Mark as Unread, 2X Voice Messages, and More

The June '18 update brought the ability to **replace** photos and videos when editing messages – in case you accidentally sent a picture of your lunch instead of the company's new logo.

As of this update, voice and video messages can be played at **2x speed** and users can **mark chats as read or unread**.

On Telegram since June 2018:

- Replace sent media
- 2x speed toggle for voice and video messages
- Mark chats as read or unread
- Share detailed contact information

- **Chat previews, text URLs and cancel sending messages on Android**

March 2018

Sticker Search, Multiple Photos and More

Since they were introduced, stickers have become a key method of communication for many Telegram users. As of this update, users can **search** for new **sticker packs**.

I'm not addicted to sending stickers. I can stop any time.

200,000,000 Monthly Active Users

Telegram has never been promoted with ads, so every user joined because they were **invited by others**. Telegram is more than a messaging app, it's the idea that everyone on the planet has a right to be **free**.

On Telegram since March 2018:

- **Sticker set search**
- **Multi-shot sending (photos)**
- **Auto-night mode on iOS** (Android since February '18)
- **Over 200,000,000 users**

February 2018

Two updates were launched on February 6, 2018.

Streaming and Auto-Night Mode on Android

This was when the apps got **streaming support**, allowing you to **view videos** without waiting for them to fully download. This update also introduced **auto-night mode** for Android users, shielding weary eyes from bright light when browsing memes in bed.

Telegram Login for Websites

Another update introduced the **Telegram Login Widget** for external websites, empowering businesses to use Telegram bots to automate anything from customer support to accepting payments or tracking shipments.

On Telegram since February 2018

- **Streaming video**
- **Auto-night mode on Android**
- **Login widget for external websites**
- **Website bots**
- **Manage connected websites via 'Active Sessions'**

January 2018

Two updates were launched on January 31, 2018.

Telegram X: Progress through Competition

Two **Telegram X** clients were unveiled as alternatives to the main iOS and Android apps – built from the ground up to be faster and easier to use, while featuring **smooth animations** and a **sleek design**. Telegram X for iOS has since become the native client on iPhones around the world.

TDLib – Build Your Own Telegram

The **Telegram Database Library** helps developers make their own versions of Telegram – and was used to create Telegram X for Android. TDLib handles network implementation, encryption, and local storage, giving third-party developers more time to focus on the fun parts of app design. All the API methods are fully documented and the code is available on **GitHub**.

Both Telegram X and the Telegram Bot API are an example of what can be done with this library.

On Telegram since January 2018

- **New app for Android: Telegram X**
- **TDLib, a library for building your own Telegram**
- **TDLib public interfaces are fully documented**
- **TDLib code is available on GitHub**

2017

December 2017

Themes, Multiple Accounts and More

Themes came to **iOS**, with simple *'Day'* and *'Night'* options. We considered adding a *'Twilight'* option but it got sabotaged by lovesick vampires and werewolves.

Android added support for **multiple accounts**, to quickly switch between up to three profiles (or personalities) without logging out.

On Telegram since December 2017

- **Multiple Accounts**
- **Quick Replies**
- **Granular Autodownload Settings**
- **Web embeds for public messages**
- **Night theme on iOS** (on Android since February '17)

November 2017

Albums, Saved Messages and Better Search

Albums and Saved Messages came to Telegram in November as a way to cut down on clutter and keep things organized. Multiple media items could be **sent as one message**, neatly arranged instead of filling up the chat with individual messages and multiple notifications.

Saved Messages brought bookmarks and personal storage, giving users the ability to forward messages there for later reference.

On Telegram since November 2017

- **Albums when sending multiple media**
- **Saved Messages**
- **Pinned Messages in Channels**
- **Select the order of photos**

October 2017

Live Locations, Media Player and Languages

October '17 expanded **location sharing** to include a **live feed**, useful for finding your friends or seeing how far they are from a destination. The interactive maps support multiple users, so your entire group can share their locations and see each other in one place.

Telegram also expanded its localization, launching the **Translations Platform** for users to refine the currently-available translations.

On Telegram since October 2017

- Live Location
- Translations Platform
- Redesigned Media Player
- Admin Badges

August 2017

Better Replies, Stickers & Invitations

Keeping track of replies and mentions was streamlined in August's update, adding the new **'@' badge** to the chat list, and **'@' button** inside chats to jump from mention to mention.

The option to set **favorite stickers** was added, while groups got the ability to choose **official sticker sets**.

On Telegram since August 2017

- Mention Badge and Button
- Favorite Stickers
- Group Sticker Sets
- Picture-in-Picture Twitch streams
- Edit photos from clipboard
- Forward to multiple people

July 2017

Disappearing Media, Your Bio & More Speed

July's update added a self-destruct timer for photos and videos to give users the option to send **disappearing media** in Private Chats.

To help say a little more about themselves, user profiles were given a **bio section**.

On Telegram since July 2017

- Disappearing Media in all one-on-one chats
- Profile Bios
- CDN Support

June 2017

Supergroups 10,000: Admin Tools & More

Groups leveled up in June '17, increasing the maximum members to 10,000 and introducing a suite of **new tools** to manage the masses.

Admin permissions became individual, allowing for **specific privileges**. So did members', meaning people could be put under **custom restrictions** for acting up.

The **Recent Actions** page was added for admins, showing a summary of all notable group activity within the past 48 hours.

On Telegram since June 2017

- Specific Admin Privileges
- Custom Member Restrictions
- Recent Actions
- Proxy Support
- Android Pay

May 2017

Three updates were released on May 18, 2017.

Video Messages and Telescope

Just in time for Summer, **video messages** were introduced to add another dimension to chats and channels.

Instant Views for Everyone & a $200K Contest

Instant View grew into a **platform** for reformatting articles from the Web into lightweight, uniform pages which instantly open in Telegram, rather than a browser. A contest to create new templates was launched.

Payments for Bots

Bot payments joined the pool party as well, so users could pay for goods and services right in the app.

On Telegram since May 2017

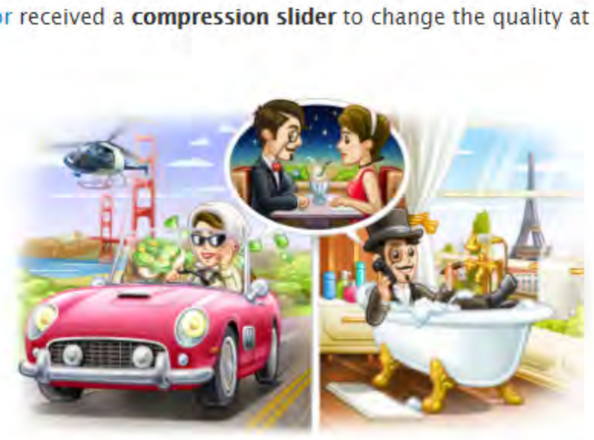
- Instant View platform and contest
- Video Messages
- Telescope
- Bot Payments

March 2017

Voice Calls: Secure, Crystal-Clear, AI-Powered

March '17 brought **voice calls** to Telegram, featuring end-to-end encryption, crisp sound and incredible speed.

The video editor received a **compression slider** to change the quality at which a video is sent.



On Telegram since March 2017

- Voice Calls
- Compression Slider

February 2017

Custom Themes

Custom themes first came to the Android phone app, along with a **theme editor**, giving users the ability to recolor individual elements and share their creations with others.

Telegram Desktop reaches version 1.0 – and it's BEAUTIFUL

Telegram Desktop, first launched in January '14, reached version **1.0**, with a **new design**, **refined animations** and **custom theme support**, in addition to all its earlier features.

The **GMail bot** made its debut as well, allowing users to condense all their communications into one platform.

Telegram for Android Wear 2.0

Telegram came to wrists with its **Android Wear 2.0** version, featuring **fully-functional messaging** capability, plus added **themes** to keep things chic.

On Telegram since February 2017

- Dark Themes and Theme Editor
- Telegram Desktop reached version 1.0
- New App: Telegram for Android Wear
- GMail Bot

January 2017

Unsend Messages, Network Usage, and More

Telegram rang in the New Year with a completely new feature – the ability to **unsend messages** sent within the past 48 hours.

To keep track of users' data consumption, **Network Usage** was added to Settings.

On Telegram since January 2017

- Delete Messages
- Network Usage statistics
- Short t.me links instead of telegram.me
- Saved scroll position when re-entering chats

2016

December 2016, Second Update

Meet the Telegraph API for Logins and Stats

The **Telegraph API** opened to the public in the final update of 2016. Publishing and organizing articles gained automated support with the new **@Telegraph bot**.

On Telegram since December 20, 2016

- Open Telegraph API
- @Telegraph bot

December 2016, First Update

Pinned Chats and IFTTT Integrations

Conversation management became easier with the option to **pin chats** to the top of the chat list.

IFTTT integration was introduced to help link Telegram to over 360 services at launch. This gives users control over their accounts on other services straight from the Telegram app – including social media, mail clients and even things like smart appliances or other devices.

On Telegram since December 7, 2016

- Pinned chats
- IFTTT integration
- Forwarding to secret chats

November 2016

Instant View, Telegraph, and Other Goodies

Instant View came in November 2016, instantly loading articles and web pages all inside the Telegram app. Added in the same update, the **Telegraph** platform is a minimalist word processor compatible with Instant View, designed for quickly publishing documents and articles.

On Telegram since November 2016

- Instant View
- Telegraph
- Jump to date
- Recent Stickers tab

> *This is where wide **full-color illustrations** for our blog posts end – or begin, depending on which direction you just came from.*

October 2016

Gaming Platform 1.0

With winter not far off, Telegram introduced a **Games** platform that allows users to play games through bots. Players get the chance to challenge friends, while creators have an easy and accessible way to bring their game ideas to life.

On Telegram since October 2016

- Gaming platform
- Games in group chats
- Game creator tools

September 2016

Photo Editor 2.0, Masks and Homemade GIFs

In September '16, the time came to improve Telegram's **Photo Editor** with fun new decorations like masks, text, stickers and drawings. Additionally, the video editor added the ability to **make GIFs** out of videos by tapping its new *'Mute'* button.

On Telegram since September 2016

- More features for the Photo Editor
- Video-to-GIF conversion
- Trending Stickers tab

August 2016

Trending Stickers, Storage and More

A **Trending Stickers** option was added so popular sticker packs can be easily found and added to one's collection. Groups received **previews** so prospective members can see what they're about and who's in them before joining the party.

On Telegram since August 2016

- Trending stickers
- Group previews
- Android camera redesign

June 2016

Drafts, Picture-in-Picture, and More

June '16 brought cross-platform **cloud drafts**, syncing your unfinished messages across devices. Since then, users have been able to start typing on one device and finish on another. For more multitasking, **Picture-in-Picture** support was added to let users work and watch simultaneously.

On Telegram since June 2016

- Cloud drafts
- Picture-in-Picture
- Android video player
- Unread message counters

May 2016

Edit Messages, New Mentions and More

Fast typers rejoiced as the ability to **edit** sent messages was introduced in all chats. Mentions were expanded to include users **without usernames**, and **Bots** were given their own space in the attachment menu for easy access.

On Telegram since May 2016

- Edit messages
- Mention users without usernames
- Bots in attachment menu

April 2016, Second Update

Instant Camera and More 3D Touch

Instant camera was added so users could snap and send photos faster, saving the extra thumb-work to make sure the moment is captured.

On Telegram since April 28, 2016

- Instant camera
- 3D Touch

April 2016, First Update

Bot Platform 2.0

Starting in April 2016, inline bots no longer require you to use a keyboard. These software-controlled friends now immediately present **clickable options** to get results with no typing required. They were also **integrated with services** like YouTube and Foursquare for easy video and location sharing.

Sharing and Previews

This update gave users a new way of **sharing** media between chats, introducing **GIF and sticker previews** to make the selection process more straightforward.

On Telegram since April 12, 2016

- New share menu
- In-app video player redesign
- Sticker previews
- GIF previews
- Inline bots 2.0 with new powers

March 2016

Supergroups 5000: Public Groups, Pinned Posts

To keep order as supergroups grew from 1,000 members to **5,000**, important announcements could be placed at the top of the screen as **pinned messages**. Additional **moderation tools** were introduced that helped prevent spam in **public** supergroups. This included the ability to report spam, ban a user outright, or simply erase all of their posts from the group.

On Telegram since March 2016

- Supergroup size increased to 5,000
- Pinned messages in supergroups
- Moderator tools for supergroups
- Public chat groups

February 2016, Second Update

Channels 2.0 and More

Channels got plenty of new features including admin **signatures** and **silent broadcasts**. It became possible to **edit** your messages in channels and supergroups.

On Telegram since February 24, 2016

- Message editing for channels
- Admin signatures in channels
- Silent messages in channels

February 2016, First Update

Voice Messages 2.0, Secret Chats 3.0 and...

Chatting got easier with **raise-to-speak** and **raise-to-listen** functionality, streamlining voice messages by giving a no-tap way to play and record. On the privacy end, users got the ability to **restrict** who can invite them to groups and channels.

On Telegram since February 12, 2016

- Raise-to-speak
- Raise-to-listen
- Invite restrictions
- Secret chats support all cloud chat features

January 2016

Introducing Inline Bots

Inline bots make it possible to use bot services in any chat without adding those bots as members – you just start any message with their username and everything that comes after becomes a query for the bot. These utilities can link Wikipedia articles, post YouTube videos or search for GIFs to entertain friends. Unfortunately, while being very good at fetch, they still can't pull a sled – even after all these years.

GIF Revolution

GIF file size was reduced by **95%**, making them load **20x** faster. Autoplay for all GIFs was enabled to show off their new-found speed.

On Telegram since January 2016

- Inline bots
- Lightweight GIFs
- GIF autoplay
- Separate GIF tab

2015

December 2015

Clearing Cache and Reordering Stickers

The space-saving option of **clearing cache** was introduced, including chat-by-chat specificity. This was coupled with a setting to choose how long to **keep media** before it gets deleted from the cache (to be re-downloaded from the cloud should you need it again).

On Telegram since December 2015

- Clear entire cache
- Clear cache for individual chats
- Reorder sticker packs

November 2015

Admins, Supergroups and More

This update added the **admin** role. Admins have the power to manage group chats and enforce rules. Just in time, as the maximum number of users for group chats grew to **1,000 people** with the addition of **supergroups**.

On Telegram since November 2015

- Admins in groups and channels
- Supergroups for 1,000 members

September 2015

Channels: Broadcasting Done Right

September '15 brought **channels**, the perfect tool for **broadcasting messages** to the masses. Channels can have unlimited followers, offer view counters for each post and only let the admins post. These one-way chats give updates the spotlight by moving the discussion elsewhere. Channels were quickly adopted in regions where freedom of speech falls short.

On Telegram since September 2015

- Channels for unlimited broadcasts
- Quick replies on iOS
- More tools for iOS photo editor

August 2015

Shared Links and Recent Searches

Telegram delivered quality of life updates in August '15, showing **shared links** on chat info pages as well as a list of **recent searches** when beginning a new one.

On Telegram since August 2015

- Shared Links tab
- Recent search results

July 2015

In-App Media Playback and Search in Chats

This update added the ability to view media, enjoy music and watch videos from external sources **directly in the app**. **Chat-specific search** was implemented as well, making it easier to find messages from a particular conversation.

On Telegram since July 2015

- In-app media playback
- Chat-specific search

June 2015

Telegram Bot Platform

In June of 2015, Telegram released the **Bot API** and created a platform for users to create and publish their own bots. Bots exist to add features to Telegram and make user's lives easier – bots can handle payments, moderate groups, fetch emails and much more.

Telegram on Apple Watch

Telegram also received **Apple Watch support**. With a flick of the wrist, users can view recent chats, reply with stickers, dictate messages and more. Anything that can't fit on the watch display gets a shortcut to open on your phone instead.

On Telegram since June 2015

- Open bot API
- Bot creation platform
- Apple Watch support

May 2015

Custom Sticker Sets

Since May '15, stickers can be uploaded and shared as **sticker sets**. All apps got a dedicated **sticker panel**. Tapping on any sticker shows its set and lets you **add** the lot.

On Telegram since May 2015

- Custom sticker sets
- Uploading stickers via the @stickers bot
- Adding and sharing sticker sets

April 2015

Migrating Existing Group Chats to Telegram

Invite links were added so users can easily bring their friends to group chats, even when those friends are lost in another messenger.

Places, Captions and more

Users gained the ability to **caption** photos, so the context and content can be sent as one message.

Active Sessions and Two-Step Verification

Telegram rolled out an essential privacy feature in April 2015 – **Two-step Verification** – adding an extra layer of protection by requiring a **password** when logging into a new device.

Link Previews

Link previews were introduced to help users see where a link leads without having to click on it and find out for themselves.

On Telegram since April 2015

- Invite links
- Photo captions
- Two-step verification
- Active sessions
- Link previews

March 2015, Second Update

Sending Files On Steroids — And More

This update brought the ability to send **multiple files** at once and **trim** videos before sending them.

On Telegram since March 25, 2015

- Multi-file sharing
- Trim videos before sending

March 2015, First Update

Reinventing Group Chats: Replies, Mentions, Hashtags and More

March '15 made messaging more convenient – Telegram added **replies**, **mentions** and **hashtags**. Replies make it easier to keep continuity in group chats, while mentions help get someone's attention and hashtags allow users to label messages for future use.

On Telegram since March 19, 2015